TUGAS AKHIR – KI141502

Implementasi *Routing Protocol* DSR pada Skenario *Mobility Random Waypoint* dengan menggunakan Propagasi Nakagami

HASBI AS SHIDDI QI
NRP 5110100058

Dosen Pembimbing
Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.
Ir. Muchammad Husni, M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya
2017

TUGAS AKHIR – KI141502

Implementasi *Routing Protocol* DSR pada Skenario *Mobility Random Waypoint* dengan menggunakan Propagasi Nakagami

HASBI AS SHIDDI QI
NRP 5110100058

Dosen Pembimbing
Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.
Ir. Muchammad Husni, M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya
2017

(Halaman ini sengaja dikosongkan)

FINAL PROJECT – KI141502

Implementation of DSR Routing Protocol in Mobility Random Waypoint Scenario using Nakagami Propagation

HASBI AS SHIDDI QI
NRP 5110100058

Advisor
Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.
Ir. Muchammad Husni, M.Kom.

DEPARTEMENT OF INFORMATICS ENGINEERING
Faculty of Information Technology
Sepuluh Nopember Intitute of Technology
Surabaya
2017

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

Implementasi *Routing Protocol* DSR pada Skenario *Mobility Random Waypoint* dengan menggunakan Propagasi Nakagami

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Arsitektur dan Jaringan Komputer
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
HASBI AS SHIDDI QI
NRP : 5110 100 058

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Dr.Eng. Radityo Anggoro, S.Kom
NIP : 19841016 200812 1 002 ..... (pembimbing)

Ir. Muchammad Husni, M.Kom.
NIP : 19600221 198403 1 001 ..... (pembimbing)

SURABAYA
JUNI 2017

(Halaman ini sengaja dikosongkan)

Implementasi *Routing Protocol* DSR pada Skenario *Mobility Random Waypoint* dengan menggunakan Propagasi Nakagami

Nama Mahasiswa	**: Hasbi As Shiddi Qi**
NRP	**: 5110100058**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing1	**: Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**
Dosen Pembimbing2	**: Ir. Muchammad Husni, M.Kom.**

ABSTRAK

Mobile Ad hoc Network (MANET) merupakan jaringan wireless yang berasal dari kumpulan mobile node yang topologinya dapat berubah dengan cepat dan kapan saja. Aspek yang penting dalam MANET adalah routing protokol dimana protokol inilah yang mengatur sistem pencarian rute paket data dalam jaringan tersebut. Ada beberapa macam protokol routing pada MANET salah satunya adalah DSR. DSR merupakan pengembangan dari AODV. Perbedaan antara AODV dengan DSR adalah jumlah rute yang ditemukan dalam setiap proses pencaria rute. Dalam Tugas Akhir ini, dilakukan penelitian terhadap kinerja DSR menggunakan Network Simulator 2 (NS-2).

Uji coba dilakukan dengan membuat pola traffic koneksi dan pola pergerakan node yang kemudian disimulasikan dengan menggunakan script tcl protokol routing DSR. Proses tersebut akan menghasilkan file otuput berupa trace file. Trace file hasil dari simulasi akan dianalisis untuk menghitung Packet Delivery Ratio (PDR), Routing Overhead (RO), dan End-to-End Delay.

Kata kunci: MANET, AODV, DSR, NS-2.

(Halaman ini sengaja dikosongkan)

Implementation of DSR Routing Protocol in Mobility Random Waypoint Scenario using Nakagami Propagation

Student's Name	**: Hasbi As Shiddi Qi**
Student's ID	**: 5110100058**
Department	**: Teknik Informatika FTIF-ITS**
Advisor1	**: Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**
Advisor2	**: Ir. Muchammad Husni, M.Kom.**

ABSTRACT

Mobile Ad hoc Network (MANET) is a wireless network that comes from a group of mobile node which topology might change quickly at any time. An important aspect in MANET is the routing protocol where protocol that regulates the packet route search system within the network. There are multiple routing protocol usable with MANET, one of them is DSR. DSR in itself is an upgrade of the existing AODV routing protocol, the main difference between the two is that the number of route found within every path discovery. In this final paper, using Network Simulator 2, the effectiveness with DSR routing protocol will be tested.

The test will be done using a connection traffic patterns and node movement pattern in which they will be simulated using a tcl routing protocol DSR script. This process will generate an output file in the form of a trace file. This particular trace file will be analyze to count for the PDR (Packet Delivery Ratio), RO (Routing Overhead), and End-to-End Delay.

Keywords: MANET, AODV, DSR, NS-2.

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Segala puji bagi Allah SWT, atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "**Implementasi *Routing Protocol* DSR pada Skenario *Mobility Random Waypoint* dengan menggunakan Propagasi Nakagami"**.

Dalam penyelesaian Tugas Akhir ini, tentunya tidak terlepas dari bantuan banyak pihak. Oleh karena itu, melalui lembar ini, penulis ingin mengucapkan terima kasih dan penghormatan sebesar-besarnya kepada :

1. Allah SWT atas limpahan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir ini dengan baik.
2. Kedua orang tua penulis, Bapak Mudloffar dan Lies Elyana, yang senantiasa memberikan kasih sayang, dukungan, serta doa yang tak terhingga kepada penulis.
3. Adik penulis, Maraya Farabi, Hana Mawaddah Fitri, keponakan penulis, Chainina Senda Arum, dan seluruh keluarga yang selalu memberikan dukungan kepada penulis.
4. Bapak Dr.Eng. Radityo Anggoro, S.Kom, M.Sc. selaku dosen pembimbing, yang telah memberikan bimbingan, dan memberikan kepercayaan, dukungan, nasihat, serta semangat kepada penulis.
5. Bapak Ir. Muchammad Husni, M.Kom. selaku dosen pembimbing kedua dan dosen wali penulis, atas bimbingan, arahan, bantuan, ide serta semua yang telah diberikan kepada penulis untuk menyelesaikan Tugas Akhir ini.
6. Bapak Dr.Eng. Darlis Herumurti, S.Kom., M.Kom. selaku Ketua Jurusan Teknik Informatika ITS.
7. Teman-teman seperjuangan Tugas Akhir Rizqi Hidayatullah, Ryeza Pattoe Marza, Adri Ramadhan, Antonio Cahyadi.
8. Bima Bahteradi, Bagus Gede Krisna, Wintang Setyo, Wiryo, Hilmy dan Seluruh teman Teknik Informatika ITS angkatan 2010 yang merupakan teman seperjuangan penulis.

9. Sahabat Sejati Penulis Adhitya Wardhana, Anantyo Dwi Cahyo, Dimas, Prasetyo, Khoirul Jamik dan Mega yang selalu memberikan semangat kepada penulis.
10. Juga tidak lupa kepada semua pihak yang belum sempat disebutkan satu per satu yang telah membantu terselesaikannya Tugas Akhir ini.

Penulis telah berusaha menyelesaikan Tugas Akhir ini sebaik mungkin, tetapi penulis mohon maaf apabila terdapat kesalahan maupun kelalaian yang penulis lakukan. Penulis mengharapkan kritik dan saran yang dapat membangun sebagai bahan perbaikan selanjutnya.

Surabaya, Juni 2017

Hasbi As Shiddi Qi

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

Bab ini membahas garis besar penyusunan Tugas Akhir yang meliputi latar belakang, tujuan pembuatan, rumusan dan batasan permasalahan, metodologi penyusunan Tugas Akhir, dan sistematika penulisan.

1.1. Latar Belakang

Perkembangan teknologi informasi dan komunikasi di era globalisasi saat ini berkembang dengan pesat. Perkembangan ini ditunjukkan dengan terciptanya berbagai macam teknologi yang membantu meningkatkan produktivitas manusia. Salah satu teknologi yang memudahkan manusia untuk saling berkomunikasi adalah *Mobile Ad hoc Network* (MANET). Pada MANET *mobile host* yang terhubung dengan *wireless* dapat bergerak bebas dan juga berperan sebagai router. Jaringan MANET adalah kumpulan dari beberapa *wireless node* yang dapat di *set-up* secara dinamis.

Dalam teknologi MANET proses pencarian rute untuk mengirimkan data dari sumber ke tujuan disebut dengan *routing*. Dalam proses *routing*, data yang berasal dari *node* sumber dikirimkan ke *node-node* lain hingga mencapai *node* tujuan. Terdapat beberapa metode *routing* pada jaringan MANET salah satunya *Dynamic Source Routing* (DSR). Protokol *routing* DSR adalah protokol *routing* yang bersifat *reactive* yaitu protokol *routing* yang hanya melakukan update jalur ketika terdapat rute baru atau ketika suatu rute terputus.

Dalam Tugas Akhir ini, akan dilakukan studi kinerja terhadap protokol *routing* DSR pada topologi jaringan MANET berdasarkan *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), dan *End-to-End Delay*.

1.2. Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana kinerja protokol *routing* DSR pada skenario *mobility random waypoint* dengan menggunakan propagasi Nakagami berdasarkan parameter berupa *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO) dan *End-to-End Delay*?

1.3. Batasan Masalah

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, diantaranya sebagai berikut:

1. Protokol *routing* yang digunakan adalah DSR.
2. Skenario uji coba dijalankan pada topologi jaringan MANET.
3. Simulasi pengujian protokol *routing* menggunakan NS-2.
4. Analisis kinerja didasarkan pada *Packet Delivery Ratio* (PDR) , *Routing Overhead* (RO), dan *End-to-End Delay*.

1.4. Tujuan dan Manfaat

Tujuan dari Tugas Akhir ini adalah menganalisis implementasi *routing protocol* DSR pada skenario *mobility random waypoint* dengan menggunakan propagasi Nakagami pada aplikasi Network Simulator 2 (NS-2).

Dengan dibuatnya Tugas Akhir ini akan memberikan sebuah manfaat untuk mengetahui kinerja propagasi Nakagami terhadap skenario *mobility random waypoint* pada *routing protocol* DSR.

1.5. Metodologi

Adapun langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir

   Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Pada proposal tersebut dijelaskan secara garis besar tentang alur pembuatan sistem.

`

2. Studi literatur

   Pada tahap ini dilakukan studi literatur mengenai *tools* dan metode yang digunakan. Literatur yang dipelajari dan digunakan meliputi buku referensi, artikel, jurnal dan dokumentasi dari internet.

3. Implementasi protokol *routing*

   Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Tahap ini merupakan tahap yang paling penting dimana bentuk awal aplikasi yang akan diimplementasikan didefinisikan. Pada tahapan ini dibuat *prototype* sistem, yang merupakan rancangan dasar dari sistem yang akan dibuat. Serta dilakukan desain suatu sistem dan desain proses-proses yang ada.

4. Uji coba dan evaluasi

   Pada tahapan ini dilakukan uji coba terhadap aplikasi yang telah dibuat. Pengujian dan evaluasi akan dilakukan dengan melihat kesesuaian dengan perencanaan. Tahap ini dimaksudkan juga untuk mengevaluasi jalannya sistem, mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

5. Penyusunan buku tugas akhir.

   Pada tahapan ini disusun buku sebagai dokumentasi dari pelaksanaan Tugas Akhir.

1.6. Sistematika Penulisan

Buku Tugas Akhir ini disusun dengan sistematika penulisan sebagai berikut:

BAB I. PENDAHULUAN

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

BAB II. TINJAUAN PUSTAKA

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang untuk mendukung pembuatan Tugas Akhir ini.

BAB III. PERANCANGAN

Bab ini berisi perancangan metode yang nantinya akan diimplementasikan dan dilakukan uji coba.

BAB IV. IMPLEMENTASI

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa implementasi skenario mobilitas *node-node* pada jaringan MANET yang dibuat menggunakan *traffic-scenario generator* dan *node-movement generator*, konfigurasi sistem dan *script* analisis yang digunakan untuk menguji performa protokol *routing*.

BAB V. UJI COBA DAN EVALUASI

Bab ini menjelaskan tahap pengujian sistem dan pengujian performa dalam skenario mobilitas *node-node* pada jaringan MANET yang dibuat.

BAB VI. PENUTUP

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan terhadap rumusan masalah yang ada dan saran untuk pengembangan lebih lanjut.

`

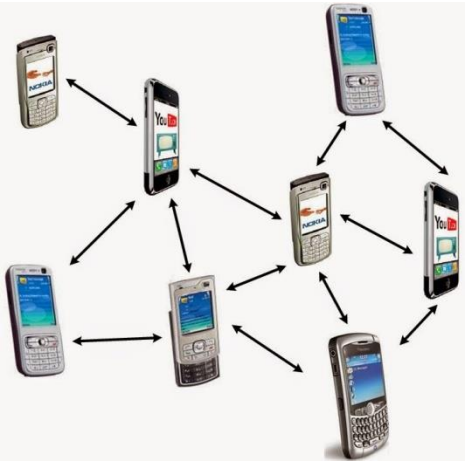
BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan tentang teori-teori yang berkaitan dengan pengimplementasian perangkat lunak. Penjelasan ini bertujuan untuk memberikan gambaran atau definisi secara umum terhadap alat, protokol *routing* serta definisi yang digunakan dalam pembuatan Tugas Akhir.

2.1. *Mobile Ad hoc Network* (MANET)

Mobile Ad hoc Network (MANET) merupakan sebuah jaringan yang terbentuk dari beberapa *node* yang bergerak bebas dan dinamis. MANET memungkinkan terjadinya komunikasi jaringan tanpa bergantung pada ketersedian infrastruktur jaringan yang tetap. Setiap *node* dalam jaringan MANET dapat bertindak sebagai *host* dan *router*. Setiap *node* dapat saling melakukan komunikasi antara yang satu dengan yang lainnya tanpa adanya *access point*.

Pada gambar 2.1 merupakan contoh penerapan jaringan MANET yang dibentuk dari sekumpulan perangkat *mobile* seperti *ponsel*. Perangkat *mobile* seperti *ponsel* harus mampu mendeteksi keberadaan perangkat lain dan melakukan pengaturan yang diperlukan untuk melakukan komunikasi dan berbagi data. Pada MANET memungkinkan perangkat untuk mempertahankan koneksi ke jaringan serta dengan mudah menambahkan dan menghapus perangkat pada jaringan. Karena pergerakan *node* yang dinamis, topologi jaringan dapat berubah dengan cepat dan tak terduga dari waktu ke waktu. Jaringan MANET bersifat desentralisasi, di mana organisasi jaringan dan pengiriman pesan harus dijalankan oleh *node* sendiri. [1]



Gambar 2.1 Jaringan MANET [1]

MANET memiliki beberapa karakteristik yaitu :

1. Topologi yang dinamis : *Node* pada MANET dapat bergerak secara bebas dan berpindah-pindah kemana saja. Topologi jaringan yang bisanya *multihop* dapat berubah secara tidak terpola.
2. Keterbatasan *bandwidth* : Link pada jaringan nirkabel memiliki kapasitas rendah daripada jaringan kabel. Selain itu, *throughput* komunikasi nirkabel seringkali lebih kecil dari tingkat transmisi maksimum radio. Hal ini dapat menyebabkan terjadinya *congestion* (kemacetan).
3. Keterbatasan energi : Semua *node* pada MANET bersifat mobile dapat dipastikan sangat mengandalkan baterai sebagai sumber energi. Sehingga diperlukan perancangan untuk optimasi energi.
4. Keterbatasan Keamanan : Jaringan nirkabel pada umumnya sangat rentan terhadap ancaman keamanan. Beberapa ancaman seperti *eavesdroping*, *spoofing* dan *denial of service* harus lebih diperhatikan. [2]

`

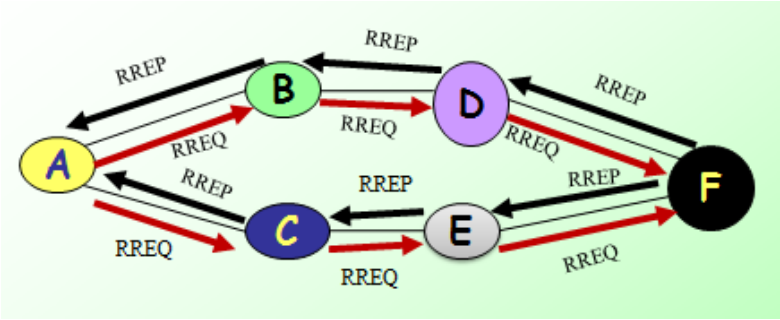
2.2. *Dynamic Source Routing* (DSR)

Protokol *Dynamic Source Routing* (DSR) [3] memiliki banyak persamaan karakteristik dengan Protokol *routing* AODV. DSR memiliki konsep bebasis vektor dan menggunakan pendekatan *multihop*. DSR juga memiliki dua fitur utama yang mirip dengan AODV yaitu *route discovery* dan *route maintenance*. Perbedaan utama antara AODV dan DSR adalah jumlah rute yang ditemukan di setiap pencarian rute atau *route discovery.*

2.2.1. *Route Discovery*

Ketika *source node* memerlukan rute untuk melakukan komunikasi dengan *destination node*, maka *source node* akan mengirimkan paket *route request* (RREQ) secara *broadcast* ke *node-node* tetangga di dalam jaringan dan menunggu *route reply* (RREP). Berbeda dengan AODV, DSR menerima semua salinan paket RREQ untuk membuat *reverse path*.

Ketika *intermediate node* menerima paket RREQ, *node* ini akan mengecek apakah ada satu atau lebih *forward path* ke *destination* yang valid. Jika ada, *node* ini akan membuat paket RREP dan mengirim kembali ke *source node* melalui *reverse path*. Jika tidak ada, maka *intermediate node* akan meneruskan paket RREQ hingga ke *destination node*, kemudian *destination node* akan membalas dengan paket RREP.



Gambar 2.2 *Route discovery* pada DSR

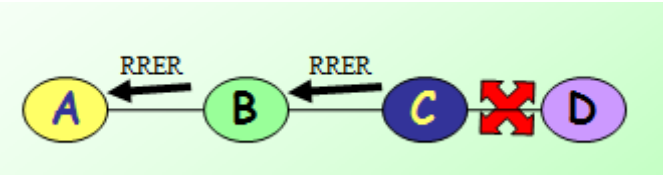
Proses *route discovery* pada DSR ditunjukkan Pada Gambar 2.2. Ketika *node* A (*source node*) hendak melakukan komunikasi terhadap *node* F (*destination node*), pertama kali *node* A mengirimkan paket RREQ secara *broadcast* ke *node-node* tetangganya yaitu ke *node* B dan *node* C. Apabila *node-node* tersebut dalam hal ini *node* B dan *node* C bukan merupakan *destination node*, maka *node* tersebut akan meneruskan paket RREQ secara *broadcast* ke *node* tetangganya tetapi tidak ke *source node*. Selain itu, *node* B dan *node* C akan melakukan *set up reverse path*. Proses tersebut berulang hingga paket RREQ tersebut diterima oleh *destination node* yatiu *node* F. Selanjutnya, *node* F akan mengiirmkan paket RREP sebagai balasan dari paket RREQ yang diterimanya. *Node-node* penerima paket RREP akan melakukan *set-up forward path*.

Pada AODV, apabila *destination node* menerima paket RREQ ganda, maka paket RREQ yang terakhir diterima *destination node* akan di-*drop*. Berbeda dengan AODV, DSR menerima semua salian paket RREQ dan akan digunakan untuk membentuk rute cadangan dari *source node* ke *destination node*. Dengan demikian, DSR akan memiliki lebih dari satu *path* (*multipath*) sehingga apabila terjadi kerusakan rute atau rute pengiriman putus saat proses pengiriman, *source node* tidak perlu lagi melakukan proses *route discovery* seperti pada AODV karena *source node* akan menggunakan rute cadangan yang telah dibentuk saat *route discovery*.

2.2.2. *Route Maintenance*

Route maintenance pada DSR adalah pengembangan sederhana yang ada pada *route maintenance* AODV. DSR juga menggunakan peket *route error* (RERR) untuk mengirimkan pesan *error*. Sebuah *node* akan mengirimkan paket RERR apabaila *path* ke *destination node* rusak. DSR memiliki beberapa rute untuk menjaga agar komunikasi tetap berlangsung yaitu ketika sebuah *node* menemui *link* yang rusak maka *node* tersebut segera memilih rute cadangan.

`



Gambar 2.3 Paket *RRER* pada DSR

Proses pengiriman paket RRER pada DSR ditunjukkan pada Gambar 2.3. Pada saat terjadi kerusakan *link* yang menghubungkan antara *node* C dengan *node* D, maka *node* C akan mengirimkan paket RRER ke *node* B. Selanjutnya, node B akan mengirimkan paket RRER ke node A. Setelah *node* A menerima paket RRER, proses komunikasi akan dilakukan kembali dengan menggunakan rute cadangan apabila masih tersedia. Apabila tidak tersedia rute cadangan, maka *node* A akan melakukan proses *route discovery*.

2.3. VirtualBox

VirtualBox adalah aplikasi *multiplatform* karena digunakan untuk meng-*install* sistem operasi di dalam sistem operasi utama pada sebuah perangkat komputer. Dengan menggunakan VirtualBox, sebuah perangkat komputer dapat menjalankan beberapa sistem operasi dalam waktu yang sama. Misalnya, pengguna dapat menjalankan Mac dan Linux di Windows, menjalankan Ubuntu di komputer dengan sistem operasi Windows, dan sebagainya.

VirtualBox didesain untuk para profesional dan pengembang di bidang Teknologi Informasi. VirtualBox dapat berjalan pada sistem operasi Windows, Mac OS X, Linux dan Oracle Solaris yang sangat ideal diaplikasikan untuk pengujian, pengembangan, dan simulasi berbagai sistem operasi pada satu mesin. [4]

Pada Tugas Akhir ini, VirtualBox digunakan untuk menjalankan sistem operasi Linux yang berisi *software* NS-2 dan juga sebagai penghubung antara sistem operasi Windows dan Linux.

2.4. Propagasi Nakagami

Propagasi Nakagami awalnya diusulkan karena cocok dengan hasil empiris untuk propagasi ionosfer gelombang pendek. Dalam komunikasi nirkabel saat ini, peran utama propagasi Nakagami dapat diringkas sebagai berikut :

- Menggambarkan amplitudo sinyal yang diterima setelah dilakukan perhitungan rata-rata *tracefile*.
- Menyalurkan amplitudo sinyal secara terdistribusi.
- Mendistribusikan transmisi sesuai dengan data empiris yang lebih baik daripada model propagasi lainnya.

2.5. Network Simulator 2 (NS-2)

Network Simulator 2 atau biasa disebut NS-2 adalah *software* simulasi yang bersifat *open source* yang dirancang khusus untuk penelitian dalam jaringan komunikasi komputer. NS-2 dikembangkan mengguanakan bahasa pemrograman C ++ dan OTcl. Simulasi kabel maupun nirkabel dan protokol dapat disimulasikan dengan menggunakan NS-2.

NS-2 merupakan *software* simulasi jaringan dengan bahasa *script* yang sederhana, sangat memudahkan peneliti untuk melakukan konfigurasi jaringan dan mengamati hasil simulasi dari NS-2. NS-2 telah mendapatkan apresiasi dari kalangan industri, akademisi, dan pemerintah sejak diperkenalkan awal tahun 1989. Berawal dari pengembangan simulator jaringan nyata oleh University of California Berkeley yang merupakan cikal bakal lahirnya Network Simulator. Pada tahun 1995 Defense Advanced Research Projects Agency (DARPA) membantu pengembangan Network Simulator melalui proyek Virtual Internetwork Terstbed (VINT). Sampai saat ini, para peneliti dan pengembang terus bekerja untuk menjadikan NS-2 lebih baik. [5] Pada Tugas Akhir ini digunakan NS-2 versi 2.35 untuk simulasi protokol *routing* DSR.

Untuk menggunakan NS-2, terlebih dahulu dilakukan proses instalasi. Proses instalasi NS-2 dimulai dengan melakukan instalasi modul-modul dependensi dari NS-2 yaitu build-essential,

`

autoconf, automake , libx11-dev, libxmu-dev dan gcc-4.4. Format baris perintah untuk installasinya adalah sebagai berikut :

```
"sudo apt-get install build-essential
autoconf automake libx11-dev libxmu-dev gcc-
4.4"
```

Setelah semua dependensi lengkap, dilakukan unduh modul seperti pada baris perintah berikut :

```
"wget http://downloads.sourceforge.net/
project/nsnam/allinone/ns-allinone-2.35/ns-
allinone.tar.gz"
```

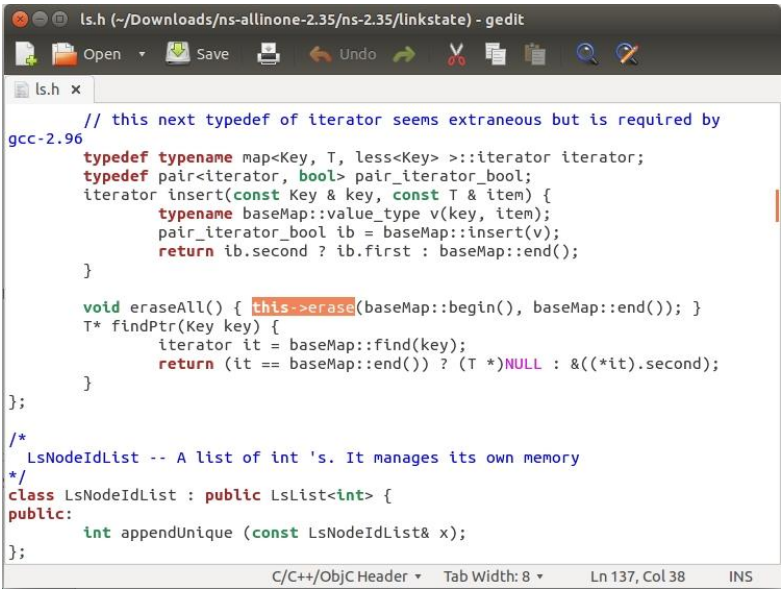
Proses ektraksi file NS-2 dapat dilakukan dengan menggunakan baris perintah sebagai berikut :

```
"tar zxvf ns-allinone-2.35.tar.gz"
```

Kemudian dilakukan proses pengubahan *script* pada ls.h yang terdapat pada *folder* /ns-allinone-2.35/ns-2.35/linkstate/ls.h yang ditunjukkan dengan menggunakan baris perintah sebagai berikut :

```
"cd /ns-allinone-2.35/ns-2.35/linkstate/"
"gedit ls.h"
```

Pada *line* ke 137, *erase* diubah menjadi *this->erase* karena jika tidak diubah, akan terjadi kegagalan saat proses instalasi NS-2. Kegagalan instalasi bisa terjadi pada beberapa sistem operasi seperti Fedora, Ubuntu atau Linux Mint. Proses tersebut yang dapat dilihat pada Gambar 2.4.



Gambar 2.4 Proses pengubahan *line of code* pada ls.h

Sampai langkah ini, proses instalasi NS-2 sudah selesai dilakukan. Untuk melakukan pengecekan apakah NS-2 telah ter-*install* dengan benar, maka dapat dilakukan dengan mengetikkan baris perintah sebagai berikut :

"ns"

apabila muncul tanda '%' pada terminal berarti NS-2 telah ter-*install* dengan benar.

2.6. *Traffic-Connection Pattern Generator*

Traffic connection pattern generator [6] merupakan *script* yang digunakan untuk membuat pola *traffic* koneksi antara *node* secara acak. *Script* tersebut digunakan pula untuk menentukan jenis *traffic* koneksi yang akan dibuat yaitu CBR atau TCP. *Traffic-Scenario Generator Script* ini terletak di direktori "~ns/indep-utils/cmu-scen-gen" dengan nama *file* cbrgen.tcl. Untuk menjalankan *script* tersebut, diperlukan baris perintah sebagai berikut :

```
"ns cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-seed
seed] [-mc connections] [-rate rate] > traffic-file"
```

Contoh baris perintah untuk menjalankan *script* cbrgen.tcl adalah sebagai berikut :

```
"ns cbrgen.tcl -type cbr -nn 50 -seed 1 -mc 2 -rate 1 > cbrfix"
```

Parameter yang digunakan pada baris perintah untuk menjalankan *script* cbrgen.tcl dijelaskan pada Tabel 2.1.

Tabel 2.1 Parameter *traffic* koneksi

No.	Parameter	Keterangan
1	-type	Jenis *traffic* yang digunakan yaitu TCP atau CBR
2	-nn	Jumlah *node* yang berkomunikasi dalam *traffic*
3	-s	Jumlah *seed*
4	-mc	Jumlah koneksi
5	-rate	Jumlah paket per detik

2.7. *Mobility Random Waypoint Generator*

Mobility random waypoint generator [6] merupakan *tool* yang digunakan untuk menghasilkan pergerakan *node-node* secara acak dalam jaringan nirkabel dan berguna untuk mendefinisikan pergerakan node dengan kecepatan gerak yang spesifik menuju suatu lokasi acak atau lokasi spesifik yang berada dalam kawasan yang telah ditentukan. Dengan *tool* ini, Sebuah *node* dapat diatur untuk berhenti sementara waktu ketika tiba di lokasi pergerakan dan kemudian *node* bergerak menuju lokasi berikutnya. *Tool* tersebut terletak di direktori "~ns/indep-utils/cmu-scen-gen/setdest" dengan nama *file* setdest. Untuk menjalan setdest, diperlukan baris perintah seperti berikut :

```
"setdest [-n num_of_nodes] [-p pausetime] [-
M maxspeed] [-t simtime] [-x maxx] [-y
maxy] > [outdir/movement-file]"
```

Contoh baris perintah untuk menjalankan *script* setdest adalah sebagai berikut :

```
"setdest -n 60 -p 10 -M 10 -t 200 -x 900 -y
900 > scenario-m10-n60-1"
```

Parameter yang digunakan pada baris perintah untuk menjalankan *script* setdest dijelaskan pada Tabel 2.2.

Tabel 2.2 Parameter baris perintah membuat 'setdest'

No.	Parameter	Keterangan
1	-n	Jumlah *node*
2	-p	Durasi ketika sebuah *node* tetap diam setelah tiba di lokasi pergerakan.
3	-M	Kecepatan maksimal sebuah *node*.
4	-t	Waktu simulasi
5	-x	Panjang maksimal area simulasi
6	-y	Lebar maksimal area simulasi

`

2.8. NS-2 *Trace File*

NS-2 *Trace file* adalah *file* berekstensi .tr yang merupakan *output* hasil *running* NS-2. *Trace file* berisi *log* tentang semua jenis pengiriman dan penerimaan paket yang terjadi selama simulasi. Di dalam *trace file* tercatat berbagai jenis paket sesuai jenis protokol *routing* yang digunakan. Tiap baris *log* ini dianalisis untuk mendapatkan performa dari sebuah protokol *routing.* Dari *Trace File* inilah yang nantinya akan dihitung *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), dan *End-to-End Delay*. Contoh *trace* pengiriman data paket pada NS-2 adalah sebagai berikut :

```
"s 2.556838879 _1_ AGT  --- 0 cbr 512 [0 0 0
0] ------- [1:0 2:0 32 0] [0] 0 3"
```

Huruf "s" di kolom pertama menandakan pengiriman paket (*send*), kolom kedua berisi waktu pengiriman paket pada detik ke 2, kolom ketiga merupakan node tempat *event* terjadi yaitu pada *node* 1, kolom ke empat berisi AGT yang menandakan pengiriman paket data, kolom ke lima merupakan tempat terjadinya *event* spesial semisal *collision*, kolom ke 6 merupakan id unik paket, kolom ke tujuh berisi tipe paket yang dikirimkan yaitu cbr, kolom ke delapan merupakan ukuran paket dalam *byte* yaitu 512.

Untuk penerimaan data paket, hampir sama dengan pengiriman data paket, yang membedakan adalah kolom pertama menggunakan huruf "r" yang menandakan bahwa paket diterima (*recieve*) dan format selanjutnya sama dengan pengiriman paket. Contoh *trace* penerimaan data paket pada NS-2 adalah sebagai berikut :

```
"r 2.556838879 _1_ RTR  --- 0 cbr 512 [0 0 0
0] ------- [1:0 2:0 32 0] [0] 0 3"
```

Contoh format untuk paket protokol *routing* DSR pada *trace file* adalah sebagai berikut :

```
"r 2.561664217 _3_ RTR --- 1 DSR 32 [0
ffffffff 1 800] ------- [1:255 2:255 32 0] 1
[1 1] [0 1 0 0->0] [0 0 0 0->0]"
```

2.9. Awk

Proses penyaringan data antara data yang akan digunakan dengan data yang akan dibuang, sering kali membutuhkan proses yang cukup kompleks. Pekerjaan seperti ini, dapat diselsaikan dengan mudah menggunakan awk. Awk adalah sebuah bahasa pemrograman yang digunakan untuk mengelola data berupa teks. Nama awk berasal dari inisial nama pengembangnya yaitu Alfred V. Aho, Peter J. Weinberger, dan Brian W. Kernighan.
Beberapa kegunaan awk adalah sebagai berikut:

- Memvalidasi Data
- Membuat laporan
- Mengelola database
- Menampilkan dokumen dan menghasilkan indeks

Selain itu, awk menyediakan fasilitas yang memudahkan untuk:

- Mengekstrak data untuk diproses
- Mengurutkan data
- Menampilkan komunikasi jaringan yang sederhana. [7]

Fungsi dasar awk adalah untuk mencari *file* per baris (atau unit teks lain) yang berisi pola tertentu. Ketika suatu baris sesuai dengan pola, awk melakukan aksi yang khusus pada baris tersebut. Awk tetap memproses baris input sedemikian hingga mencapai akhir baris input.

Program pada awk berbeda dari program di kebanyakan bahasa lain, karena program awk bersifat *data-driven* yaitu mendeskripsikan data yang dikehendaki untuk bekerja dan kemudian apa yang akan dilakukan saat data tersebut ditemukan. Kebanyakan bahasa lainnya bersifat prosedural yang artinya harus mendeskripsikannya secara detail setiap langkah program yang harus dijalankan. Ketika bekerja dengan bahasa prosedural,

`

biasanya sangat sulit untuk mendeskripsikan data yang hendak diproses oleh program. Oleh karena itu, program awk sering kali terasa lebih mudah untuk ditulis dan dibaca. [8]

Pada Tugas Akhir ini, awk digunakan untuk membuat *script* untuk menghitung *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), dan *End-to-End Delay* dari hasil *trace* NS-2.

(Halaman ini sengaja dikosongkan)

`

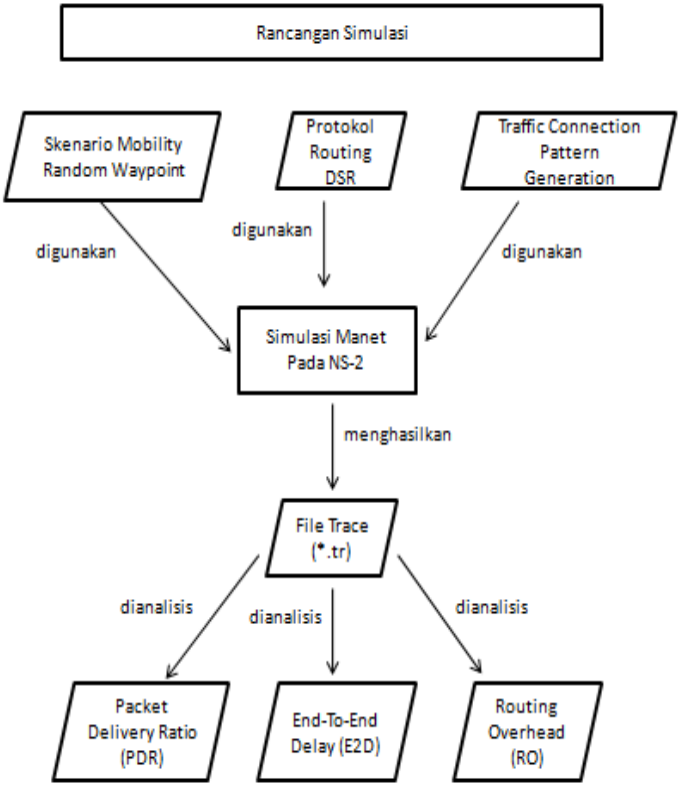
BAB III
PERANCANGAN

Pada bab ini akan dijelaskan mengenai dasar perancangan perangkat lunak yang akan dibuat dalam Tugas Akhir ini. Berawal dari deskripsi umum sistem hingga perancangan skenario, alur dan implementasinya. Sehingga bab ini secara khusus akan menjelaskan perancangan sistem yang dibuat dalam Tugas Akhir.

3.1. Deskripsi Umum

Pada Tugas Akhir ini akan dilakukan analisis tentang performa propagasi Nakagami pada MANET. Dalam pembuatan skenario MANET menggunakan *mobility random waypoint* dan telah ada pada NS-2 yaitu dengan cara men-*generate file node-movement (mobility generation)* dan membuat koneksi antar *node* menggunakan *traffic connection pattern*. Rancangan simulasi yang dibuat dapat dilihat pada Gambar 3.1.

Dalam penelitian ini, terdapat model propagasi Nakagami. Kemudian untuk simulasi skenario yang dihasilkan oleh *mobility generator random waypoint* akan dijalankan pada NS-2 menggunakan protokol *routing* DSR pada sistem operasi Linux.



Gambar 3.1 Diagram rancangan simulasi

Dalam pengujian ini, digunakan empat variasi *node* untuk simulasi protokol *routing* DSR yaitu 60, 70, 80 dan 90. Untuk kecepatan maksimal pergerakan *node* digunakan empat variasi juga yaitu 10 m/s, 15 m/s, 20 m/s dan 25 m/s. Hasil dari proses uji coba ini yaitu berupa *trace file* yang nantinya akan digunakan untuk menghitung *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO) dan *End-to-End Delay*.

`

3.2. Perancangan Skenario

Perancangan skenario uji coba diawali dengan membuat pola *traffic* koneksi antara *node* secara acak menggunakan *mobility random waypoint.* Kemudian membuat koneksi dengan menggunakan *traffic connection pattern.* Pada Tugas Akhir ini, pergerakan *node-node* menggunakan 4 variasi kecepatan maksimal yaitu 10 m/s, 15 m/s, 20 m/s dan 25 m/s.

3.2.1. Skenario *Mobility Random Waypoint*

Skenario *mobility random waypoint* dibuat dengan men-*generate file node-movement* yang telah ada pada NS-2 atau *tools*-nya biasa disebut 'setdest' yang nantinya akan menghasilkan *output* dalam bentuk .txt dan digunakan dalam *file* Tcl selama simulasi pada NS-2 sebagai bentuk pergerakan *node* yang berpindah-pindah.

Tabel 3.1 Parameter Skenario *Mobility Random Waypoint*

No.	Parameter	Spesifikasi
1	Jumlah *Node*	60, 70, 80, 90
2	Luas Lingkungan Jaringan	900 m x 900 m 500 m x 500 m
3	Waktu Simulasi	200 s
4	Kecepatan Maksimal	- 10 m/s - 15 m/s - 20 m/s - 25 m/s
5	Waktu Jeda	10 s
6	Ukuran Paket	512 bytes
7	*Rate* Paket	0.25 paket per detik
8	Jumlah maksimal koneksi	1
9	Model mobilitas yang digunakan	*Random Way Point*
10	Sumber *Traffic*	CBR

3.2.2. *Traffic-Connection Pattern*

Traffic-Connection dibuat dengan menjalankan program cbrgen.tcl yang telah ada pada NS-2 yang nantinya akan menghasilkan *output file* dan digunakan sebagai koneksi untuk menghubungkan antar *node* yang ada pada skenario selama simulasi pada NS-2. Parameter yang digunakan untuk membuat *Traffic-Connection* dapat dilihat pada Tabel 3.2.

Tabel 3.2 Parameter *Traffic-Connection Pattern*

No.	Parameter	Spesifikasi
1	Jenis *traffic*	CBR
2	Jumlah *seed*	1
3	Jumlah koneksi	2
4	Jumlah paket per detik	1
5	Jenis protokol	UDP

3.3. Perancangan Simulasi pada NS-2

Pada perancangan kode NS-2 dengan konfigurasi MANET, dilakukan penggabungan skenario pola *traffic* koneksi dan pola pergerakan *node* dengan *script* TCL yang diberikan parameter-parameter untuk membangun percobaan simulasi MANET pada NS-2. Berikut parameter simulasi perancangan sistem MANET yang dapat digunakan dapat dilihat pada Tabel 3.3.

Tabel 3.3 Parameter simulasi

No.	Parameter	Spesifikasi
1	Network simulator	NS-2 versi 2.35
2	Protokol *routing*	DSR
3	Waktu simulasi	200 s
4	Ukuran paket data	512 bytes
5	Area simulasi	900 m x 900 m 500 m x 500 m
6	Jumlah *node*	60, 70, 80, 90
7	Radius transmisi	100 m
8	*Source / destination*	Statik (*node* 1 / *node* 2)
9	Protokol MAC	IEEE 802.11

`

10	Model propagasi	Nakagami
11	Tipe antena	OmniAntenna
12	Tipe Interface Queue	CMUPriQueue
13	*Mobility model*	*Random Waypoint*
14	Tipe kanal	*Wireless channel*
15	Tipe *trace*	*Old Wireless Format Trace*

3.4. Perancangan Metrik Analisis

Metrik yang akan dianalisis pada Tugas Akhir ini adalah *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), dan *End-to-End Delay*. Penjelasannya sebagai berikut:

3.4.1. *Packet Delivery Ratio* (PDR)

PDR dihitung dari perbandingan antara paket yang dikirim dengan paket yang diterima. PDR dihitung dengan menggunakan persamaan (3.1), dimana *received* adalah banyaknya paket data yang diterima dan *sent* adalah banyaknya paket data yang dikirimkan.

$$PDR = \frac{\sum received}{\sum sent} \times 100 \quad (3.1)$$

3.4.2. *Routing Overhead* (RO)

Routing Overhead (RO) adalah jumlah paket kontrol *routing* yang ditransimisikan per data paket yang terkirim ke tujuan selama simulasi terjadi. RO dihitung berdasarkan paket *routing* yang ditransmisikan. Baris yang mengandung RO pada *trace file* ditandai dengan paket yang bertipe *send* (s) / *forward* (f) dan terdapat *header* paket dari protokol DSR.

3.4.3. *End-to-End Delay*

End-to-End Delay dihitung dari rata-rata delay antara waktu paket diterima dan waktu paket dikirim. *End-to-End Delay* dihitung dengan menggunakan persamaan (3.2), dimana $t_{received[i]}$ adalah waktu penerimaan paket dengan urutan / id ke-i, $t_{sent[i]}$ adalah waktu pengiriman paket dengan urutan / id ke-i, dan *sent* adalah banyaknya paket data yang dikirimkan.

$$End\ to\ End\ Delay = \frac{\sum_{i \leq sent}^{i=0} t_{received[i]} - t_{sent[i]}}{sent} \qquad (3.2)$$

`

BAB IV
IMPLEMENTASI

Bab ini merupakan bahasan mengenai implementasi dari perancangan sistem yang telah dijabarkan pada bab-bab sebelumnya.

4.1. Lingkungan Pembangunan Perangkat Lunak

Pembangunan perangkat lunak dilakukan pada lingkungan pengembangan sebagai berikut:

4.1.1. Lingkungan Perangkat Lunak

Adapun perangkat lunak yang digunakan untuk pengembangan sistem adalah sebagai berikut:

- Sistem Operasi Ubuntu 14.04 64-bit untuk lingkungan NS-2,
- VirtualBox untuk pengembangan aplikasi NS-2 dan juga digunakan untuk analisis, dan
- Network Simulator 2 versi 2.35.

4.1.2. Lingkungan Perangkat Keras

Spesifikasi perangkat keras yang digunakan untuk implementasi perangkat lunak Tugas Akhir adalah sebagai berikut:

- *Processor* Intel(R) Core(TM) i3 CPU @ 2.2GHz,
- Media penyimpanan sebesar 640GB, dan
- RAM sebesar 4 GB DDR3.

4.2. Implementasi Pola *Traffic* Koneksi

Untuk menghasilkan pola *traffic* koneksi, digunakan *script* cbrgen.tcl. Baris perintah *script* cbrgen.tcl yang digunakan adalah sebagai berikut :

```
"ns cbrgen.tcl -type cbr -nn 50 -seed 1
-mc 2 -rate 1 > cbrfix"
```

Setelah baris perintah script cbrgen.tcl dijalankan, maka akan menghasilkan *output file* cbrfix. Isi dari *output file* hasil dari *generate script* ditunjukkan pada Gambar 4.1.

```
# nodes: 50, max conn: 2, send rate: 1, seed: 1
# 1 connecting to 2 at time 2.5568388786897245

set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 1
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 2.5568388786897245 "$cbr_(0) start"
# 4 connecting to 5 at time 56.333118917575632
```

Gambar 4.1 Isi *file traffic* koneksi

4.3. Implementasi Pola Pergerakan *Node*

Untuk menghasilkan pola pergerakan *node*, digunakan *Node-Movement Generator* yang bernama setdest. Pada Tugas Akhir ini, pergerakan *node-node* menggunakan empat variasi kecepatan maksimal yaitu 10 m/s, 15 m/s, 20 m/s dan 25 m/s yang di simulasikan pada dua variasi luas lingkungan jaringan yaitu 900 m x 900 m dan 500 m x 500 m, dengan masing-masing dilakukan *generate* sebanyak 5 kali.

`

Contoh baris perintah setdest untuk kecepatan maksimal 10 m/s pada *node* 60 adalah sebagai berikut :

- ”setdest -v 1 -n 60 -p 10 -M 10 -t 200 -x 900 -y 900 > scenario-m10-n60-1”
- ”setdest -v 1 -n 60 -p 10 -M 10 -t 200 -x 500 -y 500 > scenario-m10-n60-1”

Untuk kecepatan maksimal 15 m/s pada *node* 70 baris perintah setdest yang digunakan adalah sebagai berikut :

- ”setdest -v 1 -n 70 -p 10 -M 15 -t 200 -x 900 -y 900 > scenario-m10-n70-1”
- ”setdest -v 1 -n 70 -p 10 -M 15 -t 200 -x 500 -y 500 > scenario-m15-n70-1”

Untuk kecepatan maksimal 20 m/s pada *node* 80 baris perintah setdest yang digunakan adalah sebagai berikut :

- ”setdest -v 1 -n 80 -p 10 -M 20 -t 200 -x 900 -y 900 > scenario-m20-n80-1”
- ”setdest -v 1 -n 80 -p 10 -M 20 -t 200 -x 500 -y 500 > scenario-m20-n80-1”

Untuk kecepatan maksimal 25 m/s pada *node* 90 baris perintah setdest yang digunakan adalah sebagai berikut :

- ”setdest -v 1 -n 90 -p 10 -M 25 -t 200 -x 900 -y 900 > scenario-m10-n90-1”
- ”setdest -v 1 -n 90 -p 10 -M 25 -t 200 -x 500 -y 500 > scenario-m25-n90-1”

Setelah perintah setdest dijalankan, maka akan menghasilkan *file output*. Gambar 4.2 menunjukkan potongan kode yang terdapat di dalam *file output* sebagai perintah untuk menentukan posisi awal *node* pada koordinat X, Y, dan Z.

```
# nodes: 60, pause: 10.00, max speed: 10.00, max x:
900.00, max y: 900.00

$node_(0) set X_ 85.149460125502
$node_(0) set Y_ 869.151048870622
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 507.271198116338
$node_(1) set Y_ 230.818556530927
$node_(1) set Z_ 0.000000000000
$node_(2) set X_ 226.247777105201
$node_(2) set Y_ 346.111459608931
$node_(2) set Z_ 0.000000000000
$node_(3) set X_ 874.737815592779
$node_(3) set Y_ 106.306603902898
$node_(3) set Z_ 0.000000000000
$node_(4) set X_ 255.923376775361
$node_(4) set Y_ 832.624086454932
$node_(4) set Z_ 0.000000000000
$node_(5) set X_ 505.099741104233
$node_(5) set Y_ 16.043233554110
$node_(5) set Z_ 0.000000000000
$node_(6) set X_ 696.761915093865
$node_(6) set Y_ 580.132913116283
$node_(6) set Z_ 0.000000000000
$node_(7) set X_ 317.599425735111
$node_(7) set Y_ 658.864135514884
$node_(7) set Z_ 0.000000000000
$node_(8) set X_ 221.594003842108
$node_(8) set Y_ 349.151731730336
$node_(8) set Z_ 0.000000000000
```

Gambar 4.2 Posisi awal *node*

`

```
$ns_ at 10.000000000000 "$node_(0) setdest
27.629667531459 192.846042223269 6.310634010614"
$ns_ at 10.000000000000 "$node_(1) setdest
39.569285798448 723.766542608720 7.018073932212"
$ns_ at 10.000000000000 "$node_(2) setdest
233.468527407724 209.046721454178 7.068608305480"
$ns_ at 10.000000000000 "$node_(3) setdest
149.149441724430 895.991444929089 8.047025267450"
$ns_ at 10.000000000000 "$node_(4) setdest
879.140318455451 30.590344677398 9.485996857539"
```

Gambar 4.3 Pergerakan *node*

Pada Gambar 4.3 menunjukkan potongan kode sebagai perintah untuk mendefinisikan pergerakan *node*. Sebagai contoh *node* 0 pada detik ke 10 mulai bergerak ke arah tujuan (27.62, 192.84) dengan kecepatan 6.31 m/s.

4.4. Implementasi Simulasi pada NS-2

Simulasi dilakukan dengan memanggil *script* protokol *routing* DSR. Dalam Tugas Akhir ini, *script* protokol *routing* DSR bernama DSR.tcl. Pada Gambar 4.4 menunjukkan potongan *script* konfigurasi awal parameter-parameter yang diberikan untuk menjalankan MANET pada NS-2. Baris pertama merupakan konfigurasi tipe *channel* yang digunakan yaitu *Wireless Channel*. Baris kedua merupakan tipe propagasi yang digunakan yaitu Nakagami. Tipe Mac yang digunakan adalah Mac 802.11. Pada *script* tersebut juga dilakukan konfigurasi tipe *queue* dari *interface*, tipe *link layer*, tipe *antenna* dan jumlah maksimum *packet* pada *interface queue.*

Selain itu juga dijelaskan pada baris berikutnya yaitu protokol *routing* yang digunakan yaitu DSR, koordinat x serta koordinat y sebesar 900, nama file tr yaitu dsr.tr, jumlah node yang digunakan yaitu 50 *node* dan yang terakhir adalah pengaturan dari baris perintah *transmission range* yang digunakan pada simulasi adalah Phy/WirelessPhy set RXThresh_, nilai yang diubah ialah RXThresh_ (*Receiver Sensitivity*

Threshold). Nilai 0.71341e-08 pada variabel tersebut menunjukkan bahwa *range* atau jangkauan yang dicapai ialah sejauh 50 meter.

```
set val(chan)   Channel/WirelessChannel;
set val(prop)   Propagation/Nakagami;
set val(netif)  Phy/WirelessPhy;
set val(mac)    Mac/802_11;
set val(ifq)    CMUPriQueue;
set val(ll)     LL;
set val(ant)    Antenna/OmniAntenna;
set opt(x)      900;
set opt(y)      900;
set val(ifqlen) 50;
set val(nn)     60;
set val(seed)   0.0;
set val(rp)     DSR;
set val(stop)   200
set val(cp)    "cbrfix"
set val(sc)    "scenario-m10-n60-1"
Phy/WirelessPhy set RXThresh_ 0.71341e-08;
```

Gambar 4.4 Konfigurasi awal parameter NS-2

```
# Initialize Global Variables
# create simulator instance
set ns_       [new Simulator]

set tracefd   [open dsr_outtrga10.tr w]

$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $opt(x)
$opt(y)
# set up topology object
set topo    [new Topography]
$topo load_flatgrid $opt(x) $opt(y)

# Create God
set god_ [create-god $val(nn)]

# Configure node
$ns_ node-config  -adhocRouting $val(adhocRouting)
\
               -llType $val(ll) \
               -macType $val(mac) \
               -ifqType $val(ifq) \
               -ifqLen $val(ifqlen) \
               -antType $val(ant) \
               -propType $val(prop) \
               -phyType $val(netif) \
              -channelType $val(chan) \
               -topoInstance $topo \
               -agentTrace ON \
               -routerTrace ON \
               -macTrace OFF \
               -movementTrace ON \

for {set i 0} {$i < $val(nn)} {incr i} {
  set node_($i) [$ns_ node]
  $node_($i) random-motion 0 ;# disable random
motion
}
```

Gambar 4.5 Konfigurasi *TraceFile* dan Pergerakan *Node* pada NS-2

Script yang ditunjukkan pada Gambar 4.5 merupakan *script* untuk pengaturan variabel global yang diawali dengan *set ns* untuk pembuatan simulator baru. *Set tracefd* merupakan pengaturan untuk nama *tracefile* berekstensi .tr yang akan dihasilkan dan disimpan.

Pada *script Set topo* merupakan pengaturan untuk objek topografi berdasarkan pada luas koordinat yang telah dikonfigurasi sebelumnya. *Create-god* dan *node-config -channelType* merupakan konfigurasi yang dilakukan pada *node-node* yang akan dibuat. Pada *create-god* dilakukan implementasi *node-node* yang akan dibuat sesuai dengan parameter pada *set-val(nn)* sedangkan pada *node-config -channelType* merupakan konfigurasi *node* sesuai dengan parameter-parameter yang telah ditambahkan sebelumnya pada Gambar 4.4 seperti tipe *link layer*, tipe mac dan tipe transmisi. Terakhir dilakukan perulangan untuk membuat pergerakan dari *node-node*. *Node-node* yang dibuat tidak dapat melakukan pergerakan secara acak karena pergerakan *node* merupakan *tracefile* yang dihasilkan oleh *mobility generator*.

Script pada Gambar 4.6 merupakan bagian akhir dari keseluruhan *script* yang digunakan untuk menginisialisasi penempatan awal *node-node* yang dibut pada skenario *node-movement (mobility generation)*, pergerakan *node* tersebut selama waktu simulasi dilakukan dan melakukan konfigurasi pengiriman paket data yang dilakukan yang nantinya dihasilkan pada *fileoutput* .tr. Pada potongan *script* tersebut, akan dipanggil *file* skenario *node-movement (mobility generation)* dan *traffic-connection pattern* kemudian pengiriman paket data dimulai pada detik ke-0 dan diberhentikan pada detik ke-200 seperti yang telah dikonfigurasi sebelumnya pada Gambar 4.4.

`

```
# Define node movement model
puts "Loading connection pattern..."
source $val(cp)

# Define traffic model
puts "Loading scenario file..."
source $val(sc)

# Define node initial position in nam

for {set i 0} {$i < $val(nn)} {incr i} {

    # 20 defines the node size in nam, must adjust
it according to your scenario
    # The function must be called after mobility
model is defined

    $ns_ initial_node_pos $node_($i) 20
}

# Tell nodes when the simulation ends
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $val(stop).0 "$node_($i) reset";
}

$ns_ at  $val(stop).0002 "puts \"NS EXITING...\" ;
$ns_ halt"

puts $tracefd "M 0.0 nn $val(nn) x $opt(x) y
$opt(y) rp $val(adhocRouting)"
puts $tracefd "M 0.0 sc $val(sc) cp $val(cp) seed
$val(seed)"
puts $tracefd "M 0.0 prop $val(prop) ant $val(ant)"

puts "Starting Simulation..."
$ns_ run
```

Gambar 4.6 Konfigurasi pengiriman paket data NS-2

Untuk menjalankan model propagasi Nakagami maka diperlukan eksekusi baris perintah sebagai berikut :

```
ns DSR.tcl
```

Pada gambar 4.7 merupakan tampilan hasil eksekusi *script* DSR.tcl menggunakan model propagasi Nakagami pada jumlah *node* 60 yang bersifat acak sesuai dengan skenario *node mobility random waypoint* yang telah di-*generate* sebelumnya.

```
hasbi@hasbi-VirtualBox:~/TA HASBI/coba/speed10$ ns DSR.tcl
num_nodes is set 60
warning: Please use -channel as shown in tcl/ex/wireless-mitf.tcl
INITIALIZE THE LIST xListHead
Loading connection pattern...
Loading scenario file...
Starting Simulation...
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 1797693134862315708145274237317043567980705
752584499659891747680315726078002853876058955863276687817154045895351438246423
213268894641827684675467035375169860499105765512820762454900903893289440758685
455133942304583236903222948165808559332123348274797826204144723168738177180919
9881250404026184124858368.0
SORTING LISTS ...DONE!
NS EXITING...
hasbi@hasbi-VirtualBox:~/TA HASBI/coba/speed10$
```

Gambar 4.7 Hasil eksekusi model propagasi Nakagami

Hasil yang didapat setelah menjalankan script dsr.tcl yaitu berupa *trace file* berbentuk file dengan ekstensi .tr. *File* .tr inilah yang akan dianalisis parameter-parameternya berupa *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO) dan *End-to-End Delay*.

4.5. Implementasi Metrik Analisis

Hasil menjalankan skenario MANET dalam NS-2 dalam bentuk *Trace File* berekstensi .tr dianalisis dengan tiga metrik yaitu *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), dan *End-to-End Delay* . Implementasi dari tiap metrik menggunakan bahasa pemrograman AWK dan dijelaskan seperti berikut.

4.5.1. *Packet Delivery Ratio* (PDR)

Proses perhitungan PDR dilakukan dengan menghitung jumlah paket data terkirim yang dilakukan oleh *node* 1 dan jumlah paket data yang diterima oleh *node* 2 pada sebuah *trace file*. Paket terkirim didapatkan dengan mencari baris yang memenuhi kondisi yaitu kolom pertama berisi huruf “s” yang berarti *send packet*, kolom ke-3 berisi angka “1” yang artinya *node* 1 melakukan pengiriman paket, kolom ke-4 dan kolom ke-7

berisi huruf "AGT" dan "cbr" yang menandakan pengiriman paket yang dilakukan adalah pengiriman paket data.

Untuk paket yang diterima didapatkan dengan mencari baris yang memenuhi kondisi yaitu kolom pertama berisi huruf "r" yang menandakan *received packet*, kolom ke-3 berisi angka "2" yang artinya node 2 menerima paket data, kolom ke-4 dan kolom ke-7 berisi huruf "AGT" dan "cbr" yang menandakan paket yang diterima adalah paket data. Perhitungan dilakukan sampai baris terakhir *trace file*, dan hasilnya adalah hasil hitung nilai PDR simulasi skenario.

Psedeucode PDR ditunjukkan pada Gambar 4.8 dan implementasinya dapat dilihat pada Lampiran.

```
BEGIN (
sent ← 0

recv ← 0
recv_id ← 0
pdr ← 0)
(if ($1 == "s" and $3 == "_1_" and $4 == "AGT" and
       $7 == "cbr" )
       sent + 1;

if ($1 == "r" and $3 == "_2_" and $4 == "AGT" and
       $7 == "cbr")
       recv +1;
END (
pdr ← ( recv / sent ) * 100
print sent
print recv
print pdr)
```

Gambar 4.8 *Pseudeucode* PDR

Contoh baris perintah untuk analisis PDR ditunjukkan sebagai berikut :

```
"awk -f pdr.awk DSR_scenario-m10-n60-1.tr"
```

Setelah baris perintah untuk analisis PDR dijalankan, maka akan menghasilkan sebuah *output* seperti yang ditunjukkan pada Gambar 4.9.

```
hasbi@hasbi-VirtualBox:~/TA HASBI/coba/speed10/tracefile$ awk -f pdr.awk DSR_sce
nario-m10-n60-1.tr
Transmitted packet (s): 191
Received packet (s): 178
Packet delivery ratio: 93.1937 %
hasbi@hasbi-VirtualBox:~/TA HASBI/coba/speed10/tracefile$
```

Gambar 4.9 Hasil *running script* pdr.awk

4.5.2. *Routing Overhead* (RO)

Baris yang mengandung *Routing Overhead* pada *trace file* ditandai dengan paket yang bertipe *send* (s) / *forward* (f) dan terdapat *header* paket dari protokol DSR. Implementasi perhitungan metrik *Routing* O*verhead* DSR dihitung dengan menggunakan beberapa kondisi-kondisi yang harus terpenuhi pada *trace file*. Kondisi-kondisi yang harus terpenuhi yatu pada kolom pertama *trace file* diawali dengan huruf "s" yang berarti *send packet* atau huruf "f" yang berarti *forward packet*, kolom ke-4 berisi huruf "RTR" yang berarti paket *routing* dan kolom ke-7 berisi huruf "DSR" yang berarti paket *routing* DSR. seperti pada Gambar 4.10. Implementasinya dapat dilihat pada Lampiran.

```
BEGIN (
rt_pkts ← 0)
(if ((($1=="s" || $1=="f") && $4 == "RTR" &&
       $7 == "DSR")
       rt_pkts + 1)
END (
print rt_pkts)
```

Gambar 4.10 *Pseudeucode Routing Overhead*

`

Trace file dianalisis dengan menggunakan baris perintah *routing overhead* yang telah dibuat. Untuk baris perintah analisis *routing overhead* adalah sebagai berikut :

```
"awk -f ro.awk DSR_scenario-m10-n60-1.tr"
```

Setelah baris perintah *routing overhead* dijalankan, akan menghasilkan sebuah *output* seperti yang ditunjukkan pada Gambar 4.11.

```
hasbi@hasbi-VirtualBox:~/TA HASBI/coba/speed10/tracefile$ awk -f ro.awk DSR_scen
ario-m10-n60-1.tr
Total number of routing packets 385
hasbi@hasbi-VirtualBox:~/TA HASBI/coba/speed10/tracefile$
```

Gambar 4.11 Hasil *running script* ro.awk

4.5.3. *End-to-End Delay* (E2D)

Perhitungan *End-to-End Delay* dilakukan dengan menghitung selisih waktu paket data terkirim yang dilakukan oleh *node* 1 dan waktu paket data diterima oleh *node* 2 di dalam sebuah *trace file*. Pencatatan waktu paket terkirim didapatkan dengan mencari baris yang memenuhi kondisi yaitu kolom pertama berisi huruf "s" yang menandakan *send packet*, kolom ke-3 berisi angka "1" yang artinya *node* yang melakukan pengiriman adalah *node* 1, kolom ke-4 dan kolom ke-7 mengandung huruf "AGT" dan "cbr" yang menunjukkan pengiriman paket yang dilakukan adalah pengiriman paket data.

Perhitungan waktu paket diterima didapatkan dengan mencari baris yang memenuhi kondisi yaitu kolom pertama berisi huruf "r" yang menandakan *received packet*, kolom ke-3 berisi angka "2" yang artinya *node* yang menerima paket adalah *node* 2, kolom ke-4 dan kolom ke-7 mengandung huruf "AGT" dan "cbr" yang menunjukkan paket yang diterima adalah paket data. Perhitungan dilakukan sampai baris terakhir *trace file*, dan dilakukan perhitungan nilai *End-to-End Delay* dengan menghitung selisih *delay* paket mulai dari pengiriman sampai paket diterima pada simulasi skenario.

Psedeucode End-to-End Delay ditunjukkan pada Gambar 4.12 dan implementasinya dapat dilihat pada Lampiran.

```
BEGIN (
for i in pkt_id
pkt_id[i] ← 0
for i in pkt_sent
       pkt_sent[i] ← 0
for i in pkt_recv
       pkt_recv[i] ← 0
delay = avg_delay ← 0
recv ← 0
recv_id ← 0)

(if ($1 == "s" and $3 == "_1_" and $4 == "AGT" and
     $7 == "cbr)
     pkt_sent[$6] ← $2

if ($1 == "r" and $3 == "_0_" and $4 == "AGT" and
    $7 == "cbr" and  recv_id != $6 )
             recv + 1
             recv_id ← $6
             pkt_recv[$6] ← $2;
END (
for i in pkt_recv
       delay += pkt_recv[i] - pkt_sent[i]
avg_delay ← delay / recv;
print  recv
print delay
print avg_delay
```

Gambar 4.12 ***Pseudeucode End-to-End Delay***

Contoh baris perintah untuk analisis *End-to-End Delay* adalah sebagai berikut :

```
"awk -f delay.awk DSR_scenario-m10-n60-1.tr"
```

Setelah baris perintah analisis *End-to-End Delay* dijalankan, akan menghasilkan sebuah *output* seperti yang ditunjukkan pada Gambar 4.13.

```
hasbi@hasbi-VirtualBox:~/TA HASBI/coba/speed10/tracefile$ awk -f delay.awk DSR_s
cenario-m10-n60-1.tr
Total Packet(s) Receive = 177
Total Delay = 6.82474 second
Average Packet Delivery Delay =  0.0385579 second
hasbi@hasbi-VirtualBox:~/TA HASBI/coba/speed10/tracefile$
```

Gambar 4.13 Hasil *running script* delay.awk

(Halaman ini sengaja dikosongkan)

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini akan dibahas mengenai pengujian dari skenario NS-2 yang telah dibuat. Pengujian fungsionalitas akan dibagi ke dalam beberapa skenario pengujian.

5.1. Lingkungan Platform

Uji coba dilakukan pada laptop dengan sistem operasi Windows yang telah terpasang perangkat lunak VirtualBox sehingga terdapat Linux dengan NS-2 terpasang di dalamnya. Untuk Spesifikasi dari laptop yang digunakan untuk uji coba ditunjukkan pada Tabel 5.1.

Tabel 5.1 Spesifikasi Komputer yang Digunakan

Komponen	Spesifikasi
CPU	*Processor* Intel(R) Core(TM) i3 CPU @ 2.2GHz
Sistem Operasi	Windows 8.1 Pro 64-bit
Memori	4 GB DDR3
Penyimpanan	640 GB

Untuk konfigurasi dari VirtualBox yang digunakan untuk uji coba ditunjukkan pada Tabel 5.2.

Tabel 5.2 Konfigurasi VirtualBox

Komponen	Konfigurasi
General	
Name	Ubuntu 14.04
Operating Siystem	Ubuntu (64-bit)
System	
Base Memory	2048 MB
Processor	1
Execution Cap	100%
Acceleration	VT-x/AMD-V, Nested Paging, PAE/NX, KVM Paravirtualization

Display	
Video Memory	128 MB
Storage	
Controller	IDE
IDE Primary Master	[Optical Drive] VboxGuestAdditions.iso (56.07 MB)
IDE Secondary Master	[Optical Drive] Empty
Controller	SATA
SATA Port 0	Ubuntu 14.04.vdi (Normal, 8.00 GB)
Audio	
Host Driver	Windows DirectSound
Controller	ICH AC97
Shared Folders	
Shared Folders	1

5.2. Kriteria Pengujian

Pengujian pada skenario yang dihasilkan oleh *mobility random waypoint* menggunakan kriteria-kriteria yang ditunjukkan pada Tabel 5.3.

Tabel 5.3 Kriteria Pengujian

Kriteria	Spesifikasi
Skenario	MANET (*Random Waypoint*)
Jumlah *Node*	60,70,80,90
Jumlah Percobaan	80 kali dengan 4 kecepatan berbeda
Posisi Awal *Node*	Acak
Pergerakan	Acak
Protokol *Routing*	DSR
Pengiriman Paket Data	Nakagami : 0 – 200 *second*

`

5.3. Analisis *Packet Delivery Ratio* (PDR)

Tracefile hasil menjalankan program skenario *mobility random waypoint* menggunakan propagasi Nakagami kemudian dianalisis nilai PDR melalui *script* pdr.awk. Hasil tiap perhitungan PDR skenario ditabulasikan dan dihitung nilai rata-rata seperti yang ditunjukkan pada Tabel 5.4 dan Tabel 5.5.

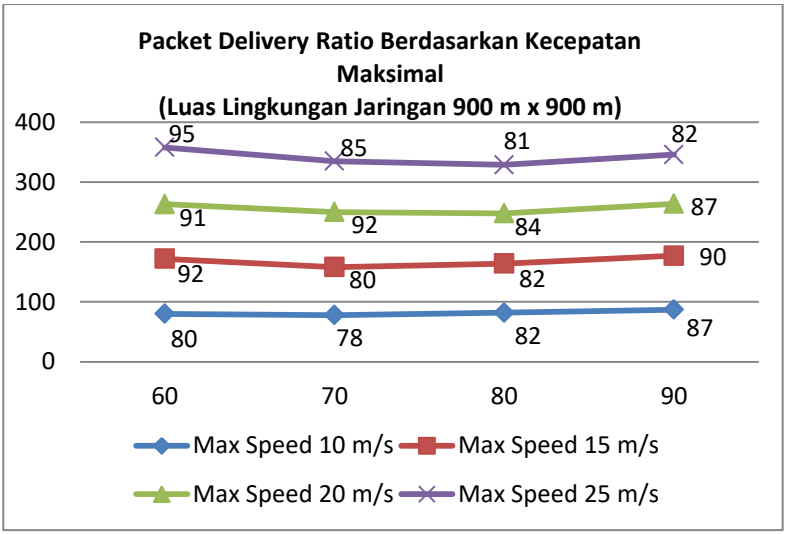
Tabel 5.4 *Packet Delivery Ratio* (PDR) DSR Luas Lingkungan Jaringan 900 m x 900 m

Max Speed (m/s)	PDR (%)			
	Node 60	Node 70	Node 80	Node 90
10	80	78	82	87
15	92	80	82	90
20	91	92	84	87
25	95	85	81	82

Tabel 5.5 *Packet Delivery Ratio* (PDR) DSR Luas Lingkungan Jaringan 500 m x 500 m

Max Speed (m/s)	PDR (%)			
	Node 60	Node 70	Node 80	Node 90
10	96	95	95	97
15	98	94	98	98
20	88	96	99	100
25	95	93	94	92

Pada Tabel 5.4 dan Tabel 5.5 menunjukkan performa PDR pada skenario *mobility random waypoint* dengan menggunakan propagasi Nakagami menghasilkan nilai yang fluktuatif ketika kecepatan maksimal perpindahan *node* bertambah.



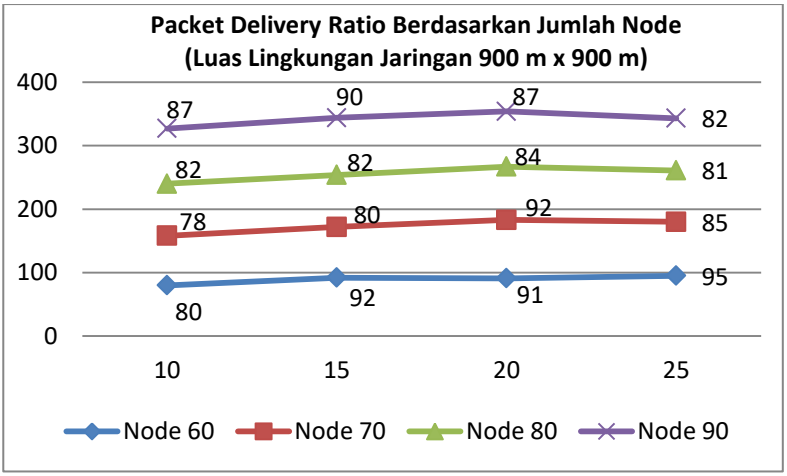
Gambar 5.1 Grafik PDR Berdasarkan Kecepatan Maksimal pada Luas Lingkungan Jaringan 900 m x 900 m

Gambar 5.1 menunjukkan grafik performa PDR berdasarkan kecepatan maksimal dengan luas lingkungan jaringan 900 m x 900 m pada skenario *mobility random waypoint* menggunakan propagasi Nakagami. Pada saat kecepatan maksimal perpindahan *node* 10 m/s dengan jumlah *node* 60, nilai PDR yang dihasilkan adalah 80%. Pada saat jumlah *node* ditambah menjadi 70, nilai PDR yang dihasilkan mengalami penurunan sebanyak 2,5% menjadi 78%. Kemudian pada saat jumlah *node* ditambah menjadi 80, nilai PDR yang dihasilkan mengalami peningkatan sebanyak 5,13% menjadi 82%. Kemudian pada saat jumlah *node* ditambah menjadi 90, nilai PDR yang dihasilkan kembali mengalami peningkatan sebanyak 6,1% menjadi 87%.

Pada saat kecepatan maksimal perpindahan *node* 15 m/s dengan jumlah *node* 60, nilai PDR yang dihasilkan adalah 92%. Pada saat jumlah *node* ditambah menjadi 70, nilai PDR yang dihasilkan mengalami penurunan sebanyak 13,04% menjadi 80%. Namun pada saat jumlah *node* ditambah menjadi 80, nilai PDR yang dihasilkan mengalami peningkatan sebanyak 2,5% menjadi 82%. Kemudian pada saat jumlah *node* ditambah menjadi 90, nilai PDR yang dihasilkan kembali mengalami peningkatan sebanyak 9,76% menjadi 90%.

Pada saat kecepatan maksimal perpindahan *node* 20 m/s dengan jumlah *node* 60, nilai PDR yang dihasilkan adalah 91%. Pada saat jumlah *node* ditambah menjadi 70, nilai PDR yang dihasilkan mengalami peningkatan sebanyak 1,1% menjadi 92%. Namun pada saat jumlah *node* ditambah menjadi 80, nilai PDR yang dihasilkan mengalami penurunan sebanayak 8,7% menjadi 78%. Kemudian pada saat jumlah *node* ditambah menjadi 90, nilai PDR yang dihasilkan mengalami peningkatan sebanyak 3,85% menjadi 87%.

Pada saat kecepatan maksimal perpindahan *node* 25 m/s dengan jumlah *node* 60, nilai PDR yang dihasilkan adalah 95%. Pada saat jumlah *node* ditambah menjadi 70, nilai PDR yang dihasilkan mengalami penurunan sebanyak 10,53% menjadi 85%. Kemudian pada saat jumlah *node* ditambah menjadi 80, nilai PDR yang dihasilkan kembali mengalami penurunan sebanyak 4,71% menjadi 81%. Namun pada saat jumlah *node* ditambah menjadi 90, nilai PDR yang dihasilkan mengalami peningkatan sebanyak 1,23% menjadi 82%.



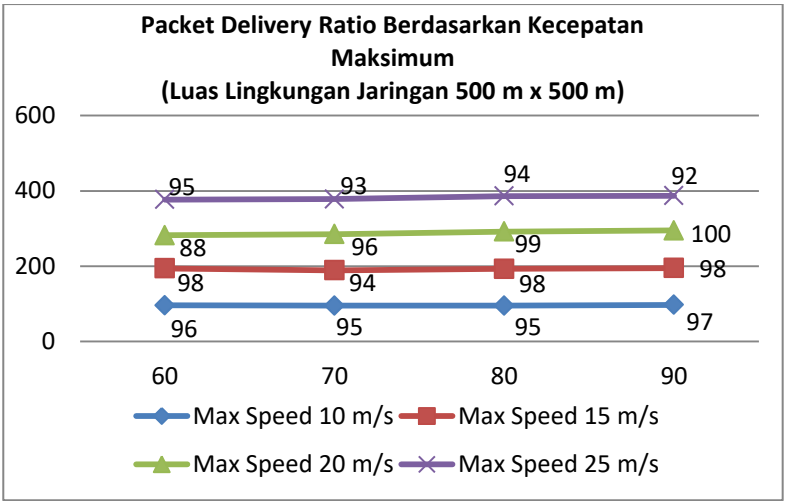
Gambar 5.2 Grafik PDR Berdasarkan Jumlah Node pada Luas Lingkungan Jaringan 900 m x 900 m

Gambar 5.2 menunjukkan grafik performa PDR berdasarkan jumlah *node* dengan luas lingkungan jaringan 900 m x 900 m pada skenario *mobility random waypoint* menggunakan propagasi Nakagami. Pada saat jumlah *node* 60 dengan kecepatan maksimal perpindahan *node* 10 m/s, nilai PDR yang dihasilkan adalah 80%. Pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 15 m/s, nilai PDR yang dihasilkan mengalami peningkatan sebanyak 15% menjadi 92%. Namun pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 20 m/s, nilai PDR yang dihasilkan mengalami penurunan sebanyak 1,09% menjadi 91%. Kemudian pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s, nilai PDR yang dihasilkan mengalami peningkatan sebanayak 4,4% menjadi 95%.

Pada saat jumlah *node* 70 dengan kecepatan maksimal perpindahan *node* 10 m/s, nilai PDR yang dihasilkan adalah 78%. Pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 15 m/s, nilai PDR yang dihasilkan mengalami peningkatan sebanyak 2,56% menjadi 80%. Kemudian pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 20 m/s, nilai PDR yang dihasilkan kembali mengalami peningkatan sebanyak 15% menjadi 92%. Namun pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25, nilai PDR yang dihasilkan mengalami penurunan sebanyak 7,61% menjadi 85%.

Pada saat jumlah *node* 80 dengan kecepatan maksimal perpindahan *node* 10 m/s, nilai PDR yang dihasilkan adalah 82%. Pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 15 m/s, nilai PDR yang dihasilkan stabil di angka 82%. Kemudian pada saat jumlah kecepatan maksimal *node* ditambah menjadi 20 m/s, nilai PDR yang dihasilkan mengalami peningkatan sebanyak 2,44% menjadi 84%. Namun pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s, nilai PDR yang dihasilkan mengalami penurunan sebanyak 3,57% menjadi 81%.

Pada saat jumlah *node* 90 dengan kecepatan maksimal perpindahan *node* 10 m/s, nilai PDR yang dihasilkan adalah 87%. Pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 15 m/s, nilai PDR yang dihasilkan mengalami peningkatan sebanyak 3,45% menjadi 90%. Namun pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 20 m/s, nilai PDR yang dihasilkan mengalami penurunan sebanyak 3,33% menjadi 87%. Penurunan nilai PDR ini juga terjadi pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s. Pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s nilai PDR yang dihasilkan mengalami penurunan sebanyak 5,75% menjadi 82%.



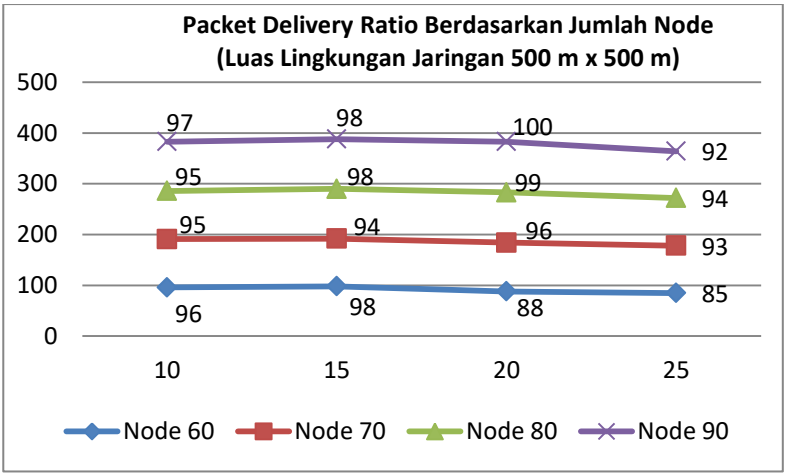
Gambar 5.3 Grafik PDR Berdasarkan Kecepatan Maksimal pada Luas Lingkungan Jaringan 500 m x 500 m

Gambar 5.3 menunjukkan grafik performa PDR berdasarkan kecepatan maksimal dengan luas lingkungan jaringan 500 m x 500 m pada skenario *mobility random waypoint* menggunakan propagasi Nakagami. Pada saat kecepatan maksimal perpindahan *node* 10 m/s dengan jumlah *node* 60, nilai PDR yang dihasilkan adalah 96%. Pada saat jumlah *node* ditambah menjadi 70, nilai PDR yang dihasilkan mengalami penurunan sebanyak 1,04% menjadi 95%. Kemudian pada saat jumlah *node* ditambah menjadi 80, nilai PDR yang dihasilkan stabil di angka 95%. Kemudian pada saat jumlah *node* ditambah menjadi 90, nilai PDR yang dihasilkan mengalami peningkatan sebanyak 2,11% menjadi 97%.

Pada saat kecepatan maksimal perpindahan *node* 15 m/s dengan jumlah *node* 60, nilai PDR yang dihasilkan adalah 98%. Pada saat jumlah *node* ditambah menjadi 70, nilai PDR yang dihasilkan mengalami penurunan sebanyak 4,08% menjadi 94%. Namun pada saat jumlah *node* ditambah menjadi 80, nilai PDR yang dihasilkan mengalami peningkatan sebanyak 4,26% menjadi 98%. Kemudian pada saat jumlah *node* ditambah menjadi 90, nilai PDR yang dihasilkan stabil di angka 98%.

Pada saat kecepatan maksimal perpindahan *node* 20 m/s dengan jumlah *node* 60, nilai PDR yang dihasilkan adalah 88%. Pada saat jumlah *node* ditambah menjadi 70, nilai PDR yang dihasilkan mengalami peningkatan sebanyak 9,09% menjadi 96%. Pada saat jumlah *node* ditambah menjadi 80, nilai PDR yang dihasilkan mengalami peningkatan sebanayak 3,13% menjadi 99%. Kemudian pada saat jumlah *node* ditambah menjadi 90, nilai PDR yang dihasilkan kembali mengalami peningkatan sebanyak 1,05% menjadi 100%.

Pada saat kecepatan maksimal perpindahan *node* 25 m/s dengan jumlah *node* 60, nilai PDR yang dihasilkan adalah 95%. Pada saat jumlah *node* ditambah menjadi 70, nilai PDR yang dihasilkan mengalami penurunan sebanyak 2,11% menjadi 93%. Kemudian pada saat jumlah *node* ditambah menjadi 80, nilai PDR yang dihasilkan kembali mengalami peningkatan sebanyak 1,08% menjadi 94%. Namun pada saat jumlah *node* ditambah menjadi 90, nilai PDR yang dihasilkan mengalami penurunan sebanyak 2,03% menjadi 92%.



Gambar 5.4 Grafik PDR Berdasarkan Jumlah Node pada Luas Lingkungan Jaringan 500 m x 500 m

Gambar 5.4 menunjukkan grafik performa PDR berdasarkan jumlah *node* dengan luas lingkungan jaringan 500 m x 500 m pada skenario *mobility random waypoint* menggunakan propagasi Nakagami. Pada saat jumlah *node* 60 dengan kecepatan maksimal perpindahan *node* 10 m/s, nilai PDR yang dihasilkan adalah 96%. Pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 15 m/s, nilai PDR yang dihasilkan mengalami peningkatan sebanyak 2,08% menjadi 98%. Namun pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 20 m/s, nilai PDR yang dihasilkan mengalami penurunan sebanyak 10,2% menjadi 88%. Kemudian pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s, nilai PDR yang dihasilkan kembali mengalami penurunan sebanayak 3,4% menjadi 85%.

Pada saat jumlah *node* 70 dengan kecepatan maksimal perpindahan *node* 10 m/s, nilai PDR yang dihasilkan adalah 95%. Pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 15 m/s, nilai PDR yang dihasilkan mengalami penurunan sebanyak 1,05% menjadi 94%. Kemudian pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 20 m/s, nilai PDR yang dihasilkan kembali mengalami peningkatan sebanyak 2,13% menjadi 96%. Namun pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25, nilai PDR yang dihasilkan mengalami penurunan sebanyak 3,13% menjadi 93%.

Pada saat jumlah *node* 80 dengan kecepatan maksimal perpindahan *node* 10 m/s, nilai PDR yang dihasilkan adalah 95%. Pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 15 m/s, nilai PDR yang dihasilkan mengalami peningkatan sebanyak 3,16% menjadi 98%. Kemudian pada saat jumlah kecepatan maksimal *node* ditambah menjadi 20 m/s, nilai PDR yang dihasilkan mengalami peningkatan sebanyak 1,02% menjadi 99%. Namun pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s, nilai PDR yang dihasilkan mengalami penurunan sebanyak 5,05% menjadi 94%.

Pada saat jumlah *node* 90 dengan kecepatan maksimal perpindahan *node* 10 m/s, nilai PDR yang dihasilkan adalah 97%. Pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 15 m/s, nilai PDR yang dihasilkan mengalami peningkatan sebanyak 1,03% menjadi 98%. Kemudian pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 20 m/s, nilai PDR yang dihasilkan mengalami peningkatan sebanyak 2,04% menjadi 100%. Penurunan nilai PDR terjadi pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s. Pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s nilai PDR yang dihasilkan mengalami penurunan sebanyak 8% menjadi 92%.

5.4. Analisis *Routing Overhead* (RO)

Trace file hasil menjalankan program DSR.tcl, dianalisis nilai RO menggunakan *script* ro.awk. Hasil tiap perhitungan RO ditabulasikan dan dihitung nilai rata-rata seperti yang ditunjukkan pada Tabel 5.6 dan Tabel 5.7.

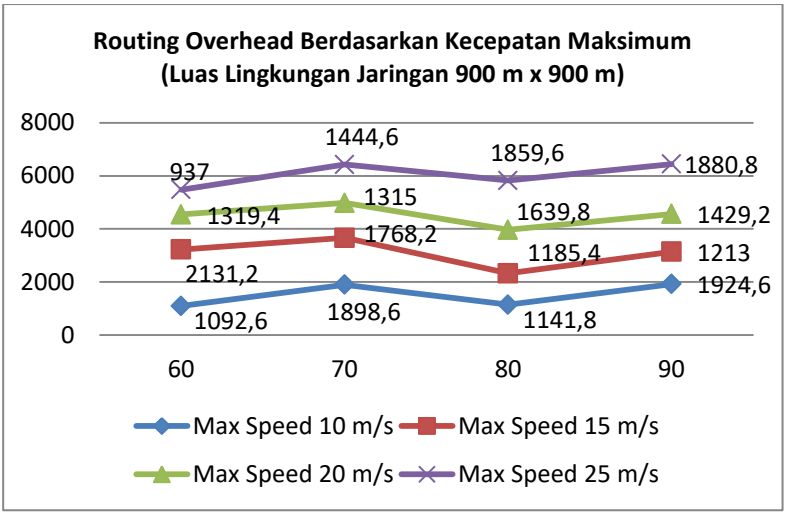
Tabel 5.6 *Routing Overhead* (RO) DSR Luas Lingkungan Jaringan 900 m x 900 m

Max Speed (m/s)	RO (Paket)			
	Node 60	*Node* 70	*Node* 80	*Node* 90
10	1092,6	1898,6	1141,8	1924,6
15	2131,2	1768,2	1185,4	1213
20	1319,4	1315	1639,8	1429,2
25	937	1444,6	1859,6	1880,8

Tabel 5.7 *Routing Overhead* (RO) DSR Luas Lingkungan Jaringan 500 m x 500 m

Max Speed (m/s)	RO (Paket)			
	Node 60	*Node* 70	*Node* 80	*Node* 90
10	628.8	350.2	561.8	537
15	328.6	758.4	413.2	593.4
20	648	465.2	887	486.6
25	407.2	817.6	785.8	958

Pada Tabel 5.6 dan Tabel 5.7 menunjukkan performa RO pada skenario *mobility random waypoint* dengan menggunakan propagasi Nakagami menghasilkan nilai yang fluktuatif ketika kecepatan maksimal perpindahan *node* bertambah.



Gambar 5.5 Grafik RO Berdasarkan Kecepatan Maksimal pada Luas Lingkungan Jaringan 900 m x 900 m

Gambar 5.5 menunjukkan grafik performa RO berdasarkan kecepatan maksimal dengan luas lingkungan jaringan 900 m x 900 m pada skenario *mobility random waypoint* menggunakan propagasi Nakagami. Pada saat kecepatan maksimal perpindahan node 10 m/s dengan jumlah *node* 60, nilai RO yang dihasilkan adalah 1092,6 paket. Pada saat jumlah *node* ditambah menjadi 70, nilai RO yang dihasilkan mengalami peningkatan sebanyak 73,77% menjadi 1898,6 paket. Namun pada saat jumlah *node* ditambah menjadi 80, nilai RO yang dihasilkan mengalami penurunan sebanyak 39,86% menjadi 1141,8 paket. Kemudian pada saat jumlah *node* ditambah menjadi 90, nilai RO yang dihasilkan mengalami peningkatan sebanyak 68,56% menjadi 1924,6 paket.

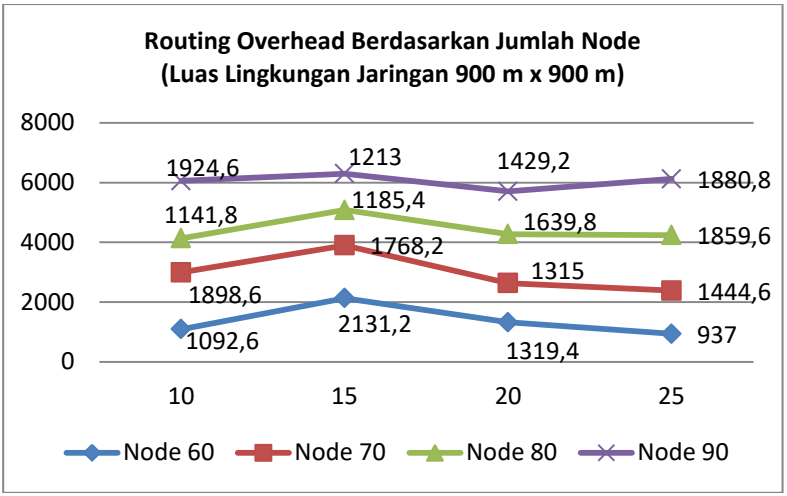
Pada saat kecepatan maksimal perpindahan *node* 15 m/s dengan jumlah *node* 60, nilai RO yang dihasilkan adalah 2131,2 paket. Pada saat jumlah *node* ditambah menjadi 70, nilai RO yang dihasilkan mengalami penurunan sebanyak 17,03% menjadi 1768,2 paket. Kemudian pada saat jumlah *node* ditambah menjadi

80, nilai RO yang dihasilkan kembali mengalami penurunan sebanyak 32,96% menjadi 1185,4 paket. Namun pada saat jumlah *node* ditambah menjadi 90, nilai RO yang dihasilkan mengalami peningkatan sebanyak 2,33% menjadi 1213 paket.

Pada saat kecepatan maksimal *node* 20 m/s dengan menggunakan jumlah *node* 60, nilai RO yang dihasilkan sebanyak 1319,4 paket. Pada saat jumlah *node* ditambah menjadi 70, nilai RO yang dihasilkan mengalami penurunan sebanyak 0,33% menjadi 1315 paket. Namun pada saat jumlah *node* ditambah menjadi 80, nilai PDR yang dihasilkan mengalami peningkatan sebanyak 24,7% menjadi 1639,8 paket. Kemudian pada saat jumlah *node* ditambah menjadi 90, nilai RO yang dihasilkan mengalami penurunan sebanyak 12,84% menjadi 1429,2 paket.

Pada saat kecepatan maksimal *node* 25 m/s dengan jumlah *node* 60, nilai RO yang dihasilkan sebanyak 937 paket. Pada saat jumlah *node* ditambah menjadi 70, nilai RO yang dihasilkan mengalami peningkatan sebanyak 54,17% menjadi 1444,6 paket. Kemudian pada saat jumlah *node* ditambah menjadi 80, nilai RO yang dihasilkan kembali mengalami peningkatan sebanyak 38,33% menjadi 1859,6 paket. Kemudian pada saat jumlah *node* ditambah menjadi 90, nilai RO yang dihasilkan mengalami peningkatan sebanyak 13,4% menjadi 1880,8 paket.

`



Gambar 5.6 Grafik RO Berdasarkan Jumlah Node pada Luas Lingkungan Jaringan 900 m x 900 m

Gambar 5.6 menunjukkan grafik performa PDR berdasarkan jumlah *node* dengan luas lingkungan jaringan 900 m x 900 m pada skenario *mobility random waypoint* menggunakan propagasi Nakagami. Pada saat jumlah *node* 60 dengan kecepatan maksimal perpindahan *node* 10 m/s, nilai RO yang dihasilkan adalah 1092,6 paket. Pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 15 m/s, nilai RO yang dihasilkan mengalami peningkatan sebanyak 95,06% menjadi 2131,2 paket. Namun pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 20 m/s, nilai RO yang dihasilkan mengalami penurunan sebanyak 38,09% menjadi 1319.4 paket. Kemudian pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s, nilai RO yang dihasilkan kembali mengalami penurunan sebanyak 28,98% menjadi 937 paket.

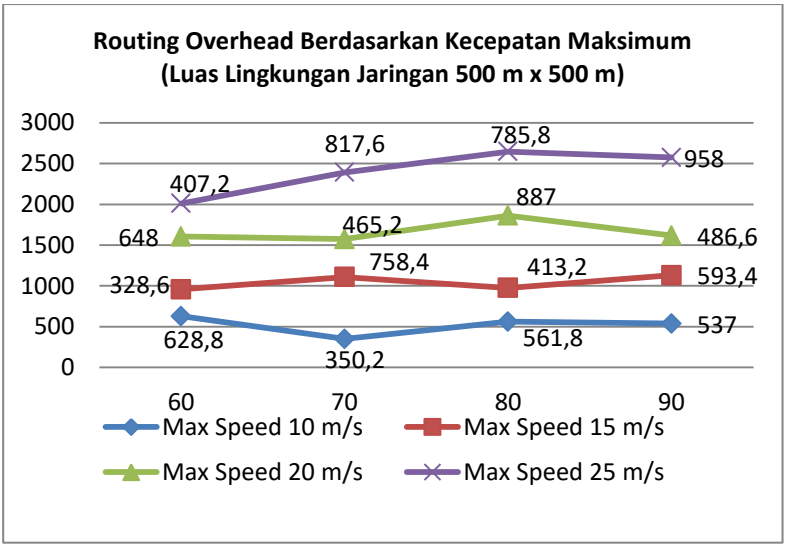
Pada saat jumlah *node* 70 dengan kecepatan maksimal perpindahan *node* 10 m/s, nilai PDR yang dihasilkan adalah 1898,6 paket. Pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 15 m/s, nilai RO yang dihasilkan

mengalami penurunan sebanyak 6,87% menjadi 1768,2 paket. Kemudian pada saat jumlah kecepatan maksimal *node* ditambah menjadi 20 m/s, nilai PDR yang dihasilkan kembali mengalami penurunan sebanyak 25,63% menjadi 1315 paket. Namun pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s, nilai RO yang dihasilkan mengalami peningkatan sebanyak 9,86% menjadi 1444,6 paket.

Pada saat jumlah *node* 80 dengan kecepatan maksimal perpindahan *node* 10 m/s, nilai RO yang dihasilkan adalah 1141,8 paket. Pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 15 m/s, nilai RO yang dihasilkan mengalami peningkatan sebanyak 3,82% menjadi 1185,4 paket. Kemudian pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 20 m/s, nilai RO yang dihasilkan kembali mengalami peningkatan sebanyak 38,33% menjadi 1639,8 paket. Peningkatan nilai RO ini juga terjadi pada saat kecepatan maksimal *node* ditambah menjadi 25 m/s. Pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s nilai RO yang dihasilkan mengalami peningkatan sebanyak 13,4% menjadi 1859,6 paket.

Pada saat jumlah *node* 90 dengan kecepatan maksimal perpindahan *node* 10 m/s, nilai RO yang dihasilkan adalah 1924,6 paket. Pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 15 m/s, nilai RO yang dihasilkan mengalami penurunan sebanyak 36,97% menjadi 1213 paket. Namun pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 20 m/s, nilai RO yang dihasilkan mengalami peningkatan sebanyak 17,82% menjadi 1429,2 paket. Kemudian pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s, nilai RO yang dihasilkan kembali mengalami peningkatan sebanyak 31,6% menjadi 1880,8 paket

`



Gambar 5.7 Grafik RO Berdasarkan Kecepatan Maksimal pada Luas Lingkungan Jaringan 500 m x 500 m

Gambar 5.7 menunjukkan grafik performa RO berdasarkan kecepatan maksimal dengan luas lingkungan jaringan 500 m x 500 m pada skenario *mobility random waypoint* menggunakan propagasi Nakagami. Pada saat kecepatan maksimal perpindahan node 10 m/s dengan jumlah *node* 60, nilai RO yang dihasilkan adalah 628,8 paket. Pada saat jumlah *node* ditambah menjadi 70, nilai RO yang dihasilkan mengalami penurunan sebanyak 44,31% menjadi 350,2 paket. Namun pada saat jumlah *node* ditambah menjadi 80, nilai RO yang dihasilkan mengalami peningkatan sebanyak 60,42% menjadi 561,8 paket. Kemudian pada saat jumlah *node* ditambah menjadi 90, nilai RO yang dihasilkan mengalami penurunan sebanyak 4,41% menjadi 537 paket.

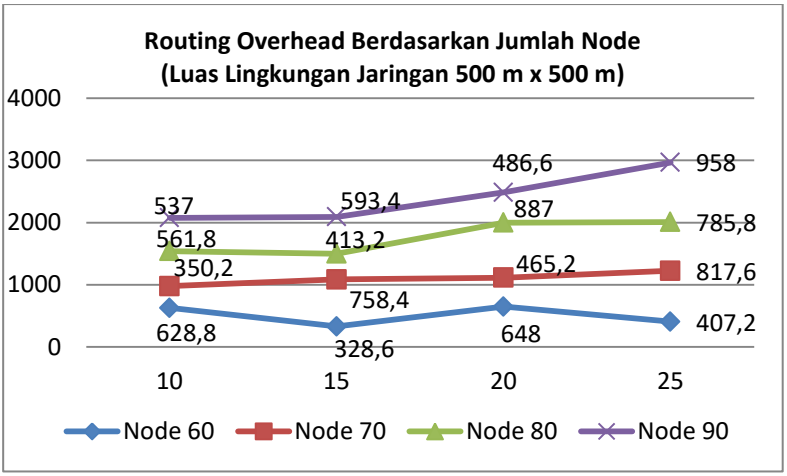
Pada saat kecepatan maksimal perpindahan *node* 15 m/s dengan jumlah *node* 60, nilai RO yang dihasilkan adalah 328,6 paket. Pada saat jumlah *node* ditambah menjadi 70, nilai RO yang dihasilkan mengalami peningkatan sebanyak 130,8% menjadi 758,4 paket. Kemudian pada saat jumlah *node* ditambah menjadi

80, nilai RO yang dihasilkan mengalami penurunan sebanyak 45,52% menjadi 413,2 paket. Namun pada saat jumlah *node* ditambah menjadi 90, nilai RO yang dihasilkan mengalami peningkatan sebanyak 18,02% menjadi 593,4 paket.

Pada saat kecepatan maksimal perpindahan *node* 20 m/s dengan menggunakan jumlah *node* 60, nilai RO yang dihasilkan sebanyak 648 paket. Pada saat jumlah *node* ditambah menjadi 70, nilai RO yang dihasilkan mengalami penurunan sebanyak 28,21% menjadi 465,2 paket. Namun pada saat jumlah *node* ditambah menjadi 80, nilai PDR yang dihasilkan mengalami peningkatan sebanyak 90,67% menjadi 887 paket. Kemudian pada saat jumlah *node* ditambah menjadi 90, nilai RO yang dihasilkan mengalami penurunan sebanyak 45,14% menjadi 486,6 paket.

Pada saat kecepatan maksimal perpindahan *node* 25 m/s dengan jumlah *node* 60, nilai RO yang dihasilkan sebanyak 407,2 paket. Pada saat jumlah *node* ditambah menjadi 70, nilai RO yang dihasilkan mengalami peningkatan sebanyak 100,79% menjadi 817,6 paket. Kemudian pada saat jumlah *node* ditambah menjadi 80, nilai RO yang dihasilkan mengalami penurunan sebanyak 3,89% menjadi 785,8 paket. Namun pada saat jumlah *node* ditambah menjadi 90, nilai RO yang dihasilkan mengalami peningkatan sebanyak 21,91% menjadi 958 paket

`



Gambar 5.8 Grafik RO Berdasarkan Jumlah Node pada Luas Lingkungan Jaringan 500 m x 500 m

Gambar 5.8 menunjukkan grafik performa RO berdasarkan jumlah *node* dengan luas lingkungan jaringan 500 m x 500 m pada skenario *mobility random waypoint* menggunakan propagasi Nakagami. Pada saat jumlah *node* 60 dengan kecepatan maksimal perpindahan *node* 10 m/s, nilai RO yang dihasilkan adalah 628,8 paket. Pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 15 m/s, nilai RO yang dihasilkan mengalami penurunan sebanyak 47,74% menjadi 328,6 paket. Namun pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 20 m/s, nilai RO yang dihasilkan mengalami peningkatan sebanyak 97,2% menjadi 648 paket. Kemudian pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s, nilai RO yang dihasilkan mengalami penurunan sebanyak 37,16% menjadi 407,2 paket.

Pada saat jumlah *node* 70 dengan kecepatan maksimal perpindahan *node* 10 m/s, nilai PDR yang dihasilkan adalah 350,2 paket. Pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 15 m/s, nilai RO yang dihasilkan mengalami peningkatan sebanyak 116,56% menjadi 758,4 paket. Kemudian

pada saat jumlah kecepatan maksimal *node* ditambah menjadi 20 m/s, nilai PDR yang dihasilkan mengalami penurunan sebanyak 38,66% menjadi 465,2 paket. Namun pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s, nilai RO yang dihasilkan mengalami peningkatan sebanyak 75,75% menjadi 817,6 paket.

Pada saat jumlah *node* 80 dengan kecepatan maksimal perpindahan *node* 10 m/s, nilai RO yang dihasilkan adalah 561,8 paket. Pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 15 m/s, nilai RO yang dihasilkan mengalami penurunan sebanyak 26,45% menjadi 413,2 paket. Kemudian pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 20 m/s, nilai RO yang dihasilkan mengalami peningkatan sebanyak 114,67% menjadi 887 paket. Namun pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s nilai RO yang dihasilkan mengalami penurunan sebanyak 11,41% menjadi 785,8 paket.

Pada saat jumlah *node* 90 dengan kecepatan maksimal perpindahan *node* 10 m/s, nilai RO yang dihasilkan adalah 537 paket. Pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 15 m/s, nilai RO yang dihasilkan mengalami peningkatan sebanyak 10,5% menjadi 593,4 paket. Namun pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 20 m/s, nilai RO yang dihasilkan mengalami penurunan sebanyak 18% menjadi 486,6 paket. Kemudian pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s, nilai RO yang dihasilkan mengalami peningkatan sebanyak 96,88% menjadi 958 paket

`

5.5. Analisis *End-to-End Delay (E2D)*

Trace file hasil menjalankan program DSR.tcl, dianalisis nilai RO menggunakan *script* ro.awk. Hasil tiap perhitungan RO ditabulasikan dan dihitung nilai rata-rata seperti yang ditunjukkan pada Tabel 5.8 dan Tabel 5.9.

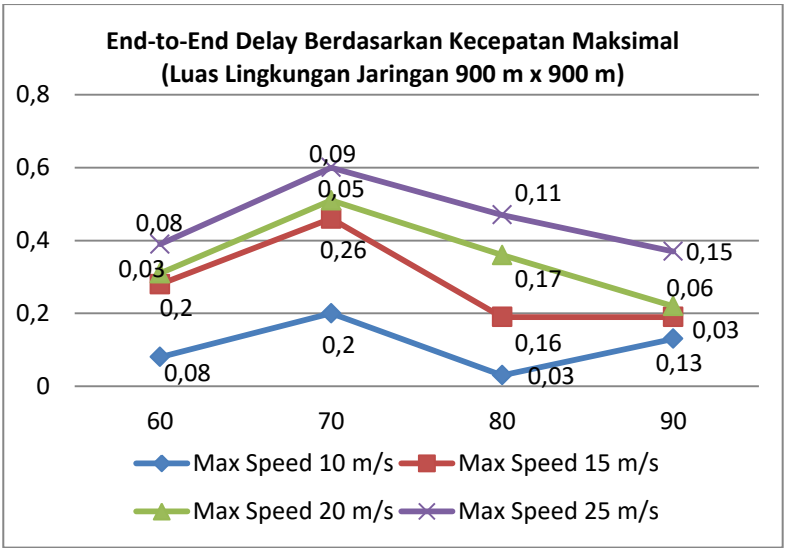
Tabel 5.8 *End-to-End Delay* DSR Luas Lingkungan Jaringan 900 m x 900 m

Max Speed (m/s)	E2D (*second*)			
	Node 60	*Node* 70	*Node* 80	*Node* 90
10	0,08	0,2	0,03	0,13
15	0,2	0,26	0,16	0,06
20	0,03	0,05	0,17	0,03
25	0,08	0,09	0,11	0,15

Tabel 5.9 *End-to-End Delay* DSR Luas Lingkungan Jaringan 500 m x 500 m

Max Speed (m/s)	E2D (*second*)			
	Node 60	*Node* 70	*Node* 80	*Node* 90
10	0.03	0.23	0.02	0.06
15	0.01	0.18	0.01	0.03
20	0.31	0.01	0.26	0.01
25	0.18	0.57	0.16	0.19

Pada Tabel 5.8 dan Tabel 5.9 menunjukkan performa *End-to-End Delay* pada skenario *mobility random waypoint* dengan menggunakan propagasi Nakagami menghasilkan nilai yang fluktuatif ketika kecepatan maksimal perpindahan *node* bertambah.



Gambar 5.9 Grafik E2D Berdasarkan Kecepatan Maksimal pada Luas Lingkungan Jaringan 900 m x 900 m

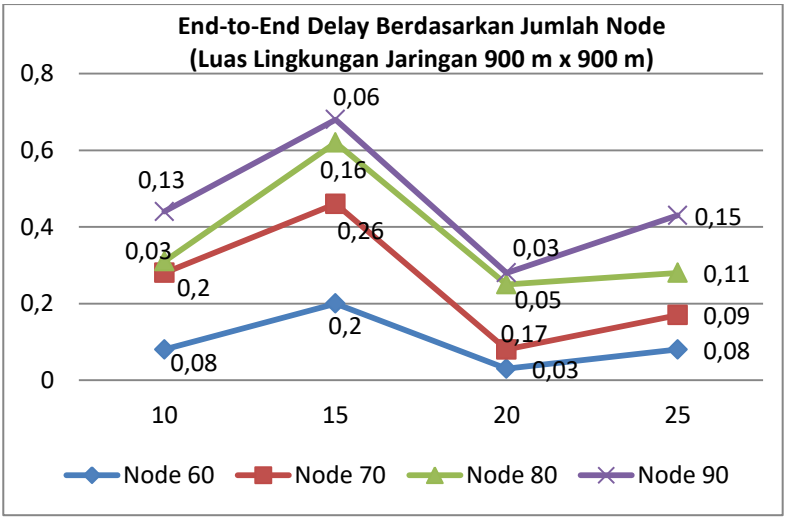
Gambar 5.9 menunjukkan grafik performa E2D kecepatan Maksimal *node* 10 m/s pada skenario *mobility random waypoint* dengan menggunakan propagasi Nakagami. Pada saat jumlah *node* 60, nilai E2D yang dihasilkan adalah 0,08 *second*. Pada saat jumlah *node* ditambah menjadi 70, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 150% menjadi 0,2 *second*. Namun pada saat jumlah *node* ditambah menjadi 80, nilai RO yang dihasilkan mengalami penurunan sebanyak 85% menjadi 0,03 *second*. Kemudian pada saat jumlah *node* ditambah menjadi 90, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 333,33% menjadi 0,13 *second*.

Pada saat kecepatan maksimal *node* 15 m/s dengan jumlah *node* 60, nilai E2D yang dihasilkan adalah 0,2 *second*. Pada saat jumlah *node* ditambah menjadi 70, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 30% menjadi 0,26 *second*. Namun pada saat jumlah *node* ditambah menjadi 80, nilai

E2D yang dihasilkan mengalami penurunan sebanyak 38,46% menjadi 0,16 *second*. Kemudian pada saat jumlah *node* ditambah menjadi 90, nilai E2D yang dihasilkan kembali mengalami penurunan sebanyak 62,5% menjadi 0,06 *second*.

Pada saat kecepatan maksimal *node* 20 m/s dengan jumlah *node* 60, nilai E2D yang dihasilkan adalah 0,03 *second*. Pada saat jumlah *node* ditambah menjadi 70, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 16,67% menjadi 0,05 *second*. Kemudian pada saat jumlah *node* ditambah menjadi 80, nilai E2D yang dihasilkan keembali mengalami peningkatan sebanyak 14,29% menjadi 0,17 *second*. Namun pada saat jumlah *node* ditambah menjadi 90, nilai E2D yang dihasilkan mengalami penurunan sebanyak 82,35% menjadi 0,03 *second*.

Pada saat kecepatan maksimal *node* 25 m/s dengan menggunakan jumlah *node* 60, nilai E2D yang dihasilkan adalah 0,08 *second*. Pada saat jumlah *node* ditambah menjadi 70, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 12,5% menjadi 0,09 *second*. Kemudian pada saat jumlah *node* ditambah menjadi 80, nilai E2D yang dihasilkan kembali mengalami peningkatan sebanyak 22,22% menjadi 0,11 *second*. Peningkatan nilai E2D juga terlihat pada saat jumlah node ditambah menjadi 90. Pada saat jumlah *node* ditambah menjadi 90, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 36,36% menjadi 0,06 *second*.



Gambar 5.10 Grafik E2D Berdasarkan Jumlah Node pada Luas Lingkungan Jaringan 900 m x 900 m

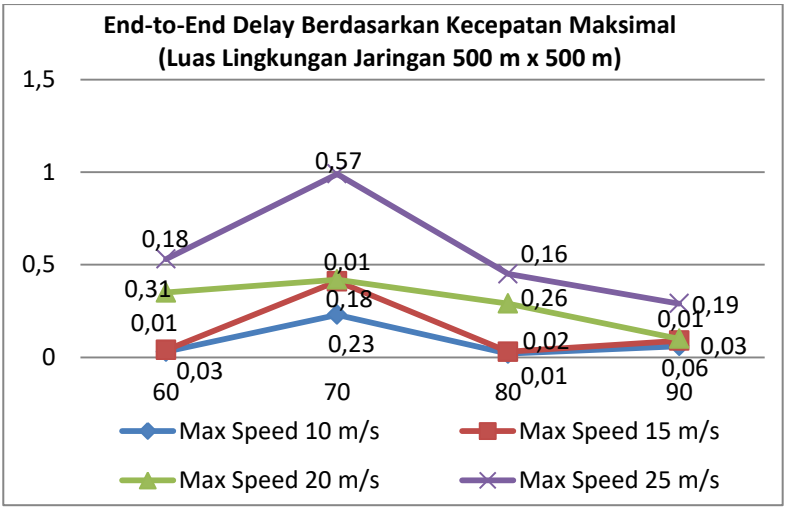
Gambar 5.10 menunjukkan grafik performa E2D *node* 60 pada skenario *mobility random waypoint* dengan menggunakan propagasi Nakagami. Pada saat kecepatan maksimal perpindahan *node* 10 m/s, nilai E2D yang dihasilkan adalah 0,08 *second*. Pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 15 m/s, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 150% menjadi 0,2 *second*. Namun pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 20 m/s, nilai E2D yang dihasilkan mengalami penurunan sebanyak 85% menjadi 0,03 *second*. Kemudian pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 166,67% menjadi 0,08 *second*.

Pada saat jumlah *node* 70 dengan kecepatan maksimal *node* 10 m/s, nilai PDR yang dihasilkan adalah 0,2 *second*. Pada saat jumlah kecepatan maksimal *node* ditambah menjadi 15 m/s, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 30%

menjadi 0,26 *second*. Namun pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 20 m/s, nilai E2D yang dihasilkan mengalami penurunan sebanyak 80,77% menjadi 0,05 *second*. Kemudian pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 80% menjadi 0,09 *second*.

Pada saat jumlah *node* 80 dengan kecepatan maksimal perpindahan *node* 10 m/s, nilai E2D yang dihasilkan adalah 0,03 *second*. Pada saat jumlah kecepatan maksimal *node* ditambah menjadi 15 m/s, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 433,33% menjadi 0,16 *second*. Kemudian pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 20 m/s, nilai PDR yang dihasilkan kembali mengalami peningkatan sebanyak 6,25% menjadi 0,17 *second*. Namun pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25, nilai E2D yang dihasilkan mengalami penurunan sebanyak 35,29% menjadi 0,11 *second*.

Pada saat jumlah *node* 90 dengan kecepatan maksimal *node* 10 m/s, nilai E2D yang dihasilkan adalah 0,13 *second*. Pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 15 m/s, nilai E2D yang dihasilkan mengalami penurunan sebanyak 53,85% menjadi 0,06 *second*. Kemudian pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 20 m/s, nilai E2D yang dihasilkan kembali mengalami penurunan sebanyak 50% menjadi 0,03 *second*. Namun pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 400% menjadi 0,15 *second*.



Gambar 5.11 Grafik E2D Berdasarkan Kecepatan Maksimal pada Luas Lingkungan Jaringan 500 m x 500 m

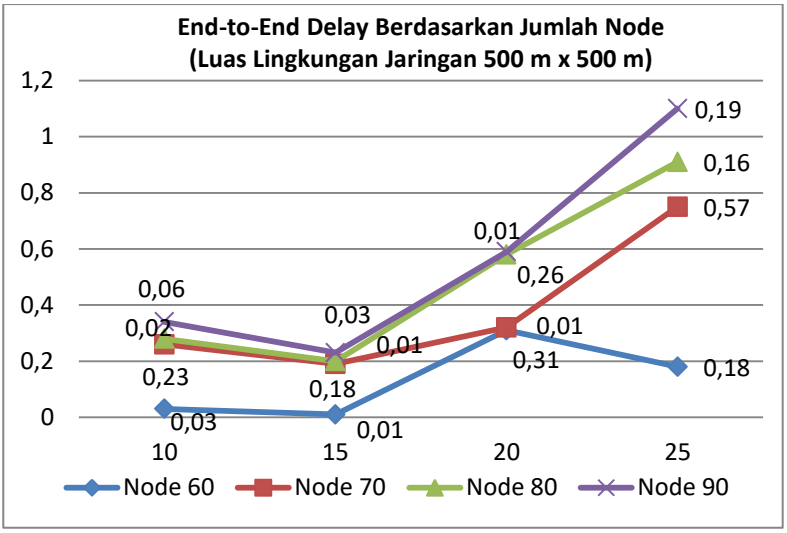
Gambar 5.11 menunjukkan grafik performa E2D kecepatan Maksimal *node* 10 m/s pada skenario *mobility random waypoint* dengan menggunakan propagasi Nakagami. Pada saat jumlah *node* 60, nilai E2D yang dihasilkan adalah 0,03 *second*. Pada saat jumlah *node* ditambah menjadi 70, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 666,67% menjadi 0,23 *second*. Namun pada saat jumlah *node* ditambah menjadi 80, nilai RO yang dihasilkan mengalami penurunan sebanyak 91,3% menjadi 0,02 *second*. Kemudian pada saat jumlah *node* ditambah menjadi 90, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 200% menjadi 0,06 *second*.

Pada saat kecepatan maksimal *node* 15 m/s dengan jumlah *node* 60, nilai E2D yang dihasilkan adalah 0,01 *second*. Pada saat jumlah *node* ditambah menjadi 70, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 1800% menjadi 0,18 *second*. Namun pada saat jumlah *node* ditambah menjadi 80, nilai E2D yang dihasilkan mengalami penurunan sebanyak 94,44%

menjadi 0,01 *second*. Kemudian pada saat jumlah *node* ditambah menjadi 90, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 200% menjadi 0,03 *second*.

Pada saat kecepatan maksimal *node* 20 m/s dengan jumlah *node* 60, nilai E2D yang dihasilkan adalah 0,31 *second*. Pada saat jumlah *node* ditambah menjadi 70, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 16,67% menjadi 0,01 *second*. Kemudian pada saat jumlah *node* ditambah menjadi 80, nilai E2D yang dihasilkan keembali mengalami peningkatan sebanyak 14,29% menjadi 0,26 *second*. Namun pada saat jumlah *node* ditambah menjadi 90, nilai E2D yang dihasilkan mengalami penurunan sebanyak 96,15% menjadi 0,01 *second*.

Pada saat kecepatan maksimal *node* 25 m/s dengan menggunakan jumlah *node* 60, nilai E2D yang dihasilkan adalah 0,18 *second*. Pada saat jumlah *node* ditambah menjadi 70, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 216,67% menjadi 0,57 *second*. Kemudian pada saat jumlah *node* ditambah menjadi 80, nilai E2D yang dihasilkan mengalami penurunan sebanyak 71,93% menjadi 0,16 *second*. Namun pada saat jumlah *node* ditambah menjadi 90, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 18,75% menjadi 0,19 *second*.



Gambar 5.12 Grafik E2D Berdasarkan Jumlah Node pada Luas Lingkungan Jaringan 500 m x 500 m

Gambar 5.12 menunjukkan grafik performa E2D *node* 60 pada skenario *mobility random waypoint* dengan menggunakan propagasi Nakagami. Pada saat kecepatan maksimal perpindahan *node* 10 m/s, nilai E2D yang dihasilkan adalah 0,03 *second*. Pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 15 m/s, nilai E2D yang dihasilkan mengalami penurunan sebanyak 66,67% menjadi 0,01 *second*. Namun pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 20 m/s, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 3000% menjadi 0,31 *second*. Kemudian pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s, nilai E2D yang dihasilkan mengalami penurunan sebanyak 41,94% menjadi 0,18 *second*.

Pada saat jumlah *node* 70 dengan kecepatan maksimal *node* 10 m/s, nilai PDR yang dihasilkan adalah 0,23 *second*. Pada saat jumlah kecepatan maksimal *node* ditambah menjadi 15 m/s, nilai E2D yang dihasilkan mengalami penurunan sebanyak 21,74%

menjadi 0,18 *second*. Kemudian pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 20 m/s, nilai E2D yang dihasilkan kembali mengalami penurunan sebanyak 94,44% menjadi 0,01 *second*. Kemudian pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 5600% menjadi 0,57 *second*.

Pada saat jumlah *node* 80 dengan kecepatan maksimal perpindahan *node* 10 m/s, nilai E2D yang dihasilkan adalah 0,02 *second*. Pada saat jumlah kecepatan maksimal *node* ditambah menjadi 15 m/s, nilai E2D yang dihasilkan mengalami penurunan sebanyak 50% menjadi 0,01 *second*. Kemudian pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 20 m/s, nilai PDR yang dihasilkan mengalami peningkatan sebanyak 2500% menjadi 0,26 *second*. Namun pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25, nilai E2D yang dihasilkan mengalami penurunan sebanyak 38,46% menjadi 0,16 *second*.

Pada saat jumlah *node* 90 dengan kecepatan maksimal *node* 10 m/s, nilai E2D yang dihasilkan adalah 0,06 *second*. Pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 15 m/s, nilai E2D yang dihasilkan mengalami penurunan sebanyak 50% menjadi 0,03 *second*. Kemudian pada saat jumlah kecepatan maksimal perpindahan *node* ditambah menjadi 20 m/s, nilai E2D yang dihasilkan kembali mengalami penurunan sebanyak 66,67% menjadi 0,01 *second*. Namun pada saat kecepatan maksimal perpindahan *node* ditambah menjadi 25 m/s, nilai E2D yang dihasilkan mengalami peningkatan sebanyak 1800% menjadi 0,19 *second*.

(Halaman ini sengaja dikosongkan)

BAB VI
PENUTUP

Pada bab ini akan diberikan kesimpulan yang diambil selama pengerjaan Tugas Akhir ini serta saran-saran tentang pengembangan yang dapat dilakukan terhadap Tugas Akhir ini di masa yang akan datang.

6.1. Kesimpulan

Kesimpulan yang dapat diambil dalam Tugas Akhir ini adalah sebagai berikut:

1. Hal – hal yang dapat mempengaruhi nilai PDR, E2D dan RO yang dihasilkan dari model propagasi Nakagami adalah:
   - Jumlah *node* yang digunakan dalam simulasi.
   - Kecepatan maksimal perpindahan *node.*
   - Luas lingkungan jaringan.

2. Implementasi *routing protocol* DSR pada skenari*o mobility random waypoint* dengan menggunakan propagasi Nakagami memiliki performa sebagai berikut :
   - Performa *Packet Delivery Ratio* yang dihasilkan semakin menurun secara stabil pada luas lingkungan jaringan 900 m x 900 m saat jumlah *node* 90, yaitu dengan rentang nilai 90% menjadi 82% pada saat kecepatan maksimal perpindahan *node* sebesar 15 m/s hingga 25 m/s.
   - Performa *End-to-End Delay* yang dihasilkan memiliki nilai *delay* fluktuatif pada luas lingkungan jaringan 500 m x 500 m saat jumlah *node* 60, yaitu mengalami penurunan pada saat kecepatan maksimal perpindahan *node* sebesar 15 m/s dan 25 m/s, namun pada saat kecepatan perpindahan *node* sebesar 20 m/s nilai *delay* mengalami peningkatan.

- Performa *Routing Overhead* yang dihasilkan semakin bertambah secara stabil pada luas lingkungan jaringan 900 m x 900 m saat *node* 80, yaitu dengan rentang nilai 1185,4 paket menjadi 1859,6 paket pada saat kecepatan maksimal perpindahan *node* sebesar 15 m/s hingga 25 m/s.

6.2. Saran

Dalam pengerjaan Tugas Akhir ini terdapat beberapa saran untuk perbaikan serta pengembangan sistem yang telah dikerjakan sebagai berikut:

1. Dapat dilakukan percobaan pada lingkungan VANET untuk penerapan model transmisi Nakagami.
2. Perlu dikembangkan penelitian menggunakan skenario yang lebih riil seperti menggunakan *Simulation of Urban Mobility* (SUMO) atau simulator lainnya.

`

DAFTAR PUSTAKA

[1] Pankaj Rohal dan Ruchika Dahiya, "Study and Analysis of Throughput, Delay and Packet Delivery Ratio in Manet Based DSR Routing Protocols," *International Journal For Advance Research In Engineering And Technology*, vol. I, no. 2, 2013.

[2] Swarupkumar dan Anil, "A Protocol for Reducing Routing Overhead in Mobile Ad Hoc Networks," *Journal of Computer Science & Engineering Technology*, 2014.

[3] M. K. Marina dan S. R. Das, "Ad hoc on-demand multipath distance vector routing," *WIRELESS COMMUNICATIONS AND MOBILE COMPUTING*, 2006.

[4] "VirtualBox," Oracle, [Online]. Available: https://www.virtualbox.org/manual/ch01.html. [Diakses 18 mei 2017].

[5] T. Issariyakul dan E. Hossain, Introduction to Network Simulator NS2, New York: Springer, 2012.

[6] "Information Sciences Institute," [Online]. Available: http://www.isi.edu/nsnam/ns/tutorial/nsscript7.html. [Diakses 18 mei 2017].

[7] "GNU Operating System," [Online]. Available: http://www.gnu.org/software/gawk/manual/gawk.html. [Diakses 18 mei 2017].

[8] "Bengkel Ubuntu," 8 November 2014. [Online]. Available: http://bengkelubuntu.org/teks/awk/Praktikum%2001%20-%20Berkenalan%20dengan%20AWK.pdf. [Diakses 18 mei 2017].

(Halaman ini sengaja dikosongkan)

LAMPIRAN

```
1   #
2   # nodes: 60, pause: 10.00, max speed: 10.00,
3   max x: 900.00, max y: 900.00
4   #
5   $node_(0) set X_ 85.149460125502
6   $node_(0) set Y_ 869.151048870622
7   $node_(0) set Z_ 0.000000000000
8   $node_(1) set X_ 507.271198116338
9   $node_(1) set Y_ 230.818556530927
10  $node_(1) set Z_ 0.000000000000
11  $node_(2) set X_ 226.247777105201
12  $node_(2) set Y_ 346.111459608931
13  $node_(2) set Z_ 0.000000000000
14  $node_(3) set X_ 874.737815592779
15  $node_(3) set Y_ 106.306603902898
16  $node_(3) set Z_ 0.000000000000
17  $node_(4) set X_ 255.923376775361
18  $node_(4) set Y_ 832.624086454932
19  $node_(4) set Z_ 0.000000000000
20  $node_(5) set X_ 505.099741104233
21  $node_(5) set Y_ 16.043233554110
22  $node_(5) set Z_ 0.000000000000
23  $node_(6) set X_ 696.761915093865
24  $node_(6) set Y_ 580.132913116283
25  $node_(6) set Z_ 0.000000000000
26  $node_(7) set X_ 317.599425735111
27  $node_(7) set Y_ 658.864135514884
28  $node_(7) set Z_ 0.000000000000
29  $node_(8) set X_ 221.594003842108
30  $node_(8) set Y_ 349.151731730336
31  $node_(8) set Z_ 0.000000000000
32  $node_(9) set X_ 324.256733373134
33  $node_(9) set Y_ 251.180178240286
34  $node_(9) set Z_ 0.000000000000
35  $node_(10) set X_ 425.780222858835
36  $node_(10) set Y_ 284.420249461991
37  $node_(10) set Z_ 0.000000000000
38  $node_(11) set X_ 894.892546147492
39  $node_(11) set Y_ 3.481007908482
40  $node_(11) set Z_ 0.000000000000
41  $node_(12) set X_ 864.468958184483
42  $node_(12) set Y_ 357.401039460538
```

```
43  $node_(12) set Z_ 0.000000000000
44  $node_(13) set X_ 624.056033930661
45  $node_(13) set Y_ 758.358136689871
46  $node_(13) set Z_ 0.000000000000
47  $node_(14) set X_ 106.253930134889
48  $node_(14) set Y_ 804.762385620404
49  $node_(14) set Z_ 0.000000000000
50  $node_(15) set X_ 446.862361172068
51  $node_(15) set Y_ 531.893472381365
52  $node_(15) set Z_ 0.000000000000
53  $node_(16) set X_ 446.832515727999
54  $node_(16) set Y_ 432.959582064206
55  $node_(16) set Z_ 0.000000000000
56  $node_(17) set X_ 487.535817759915
57  $node_(17) set Y_ 489.431380015269
58  $node_(17) set Z_ 0.000000000000
59  $node_(18) set X_ 529.426047001178
60  $node_(18) set Y_ 341.296352513517
61  $node_(18) set Z_ 0.000000000000
62  $node_(19) set X_ 10.970861320835
63  $node_(19) set Y_ 508.370593735269
64  $node_(19) set Z_ 0.000000000000
65  $node_(20) set X_ 519.723256002748
66  $node_(20) set Y_ 497.146575457064
67  $node_(20) set Z_ 0.000000000000
68  $node_(21) set X_ 261.974719513846
69  $node_(21) set Y_ 705.341066445909
70  $node_(21) set Z_ 0.000000000000
71  $node_(22) set X_ 474.203961746400
72  $node_(22) set Y_ 339.403253280529
73  $node_(22) set Z_ 0.000000000000
74  $node_(23) set X_ 399.648257118379
75  $node_(23) set Y_ 867.713739323536
76  $node_(23) set Z_ 0.000000000000
77  $node_(24) set X_ 483.758597779504
78  $node_(24) set Y_ 594.725670200526
79  $node_(24) set Z_ 0.000000000000
80  $node_(25) set X_ 173.108286644920
81  $node_(25) set Y_ 655.569729525248
82  $node_(25) set Z_ 0.000000000000
83  $node_(26) set X_ 801.304519728604
84  $node_(26) set Y_ 400.394380856778
85  $node_(26) set Z_ 0.000000000000
86  $node_(27) set X_ 681.525002642814
```

```
87  $node_(27) set Y_ 540.901419498840
88  $node_(27) set Z_ 0.000000000000
89  $node_(28) set X_ 21.023588830816
90  $node_(28) set Y_ 56.466685408976
91  $node_(28) set Z_ 0.000000000000
92  $node_(29) set X_ 589.113167891172
93  $node_(29) set Y_ 651.939833444423
94  $node_(29) set Z_ 0.000000000000
95  $node_(30) set X_ 124.391997110456
96  $node_(30) set Y_ 465.559969245566
97  $node_(30) set Z_ 0.000000000000
98  ...
99  $god_ set-dist 0 1 4
100 $god_ set-dist 0 2 3
101 $god_ set-dist 0 3 7
102 $god_ set-dist 0 4 1
103 $god_ set-dist 0 5 5
104 $god_ set-dist 0 6 3
105 $god_ set-dist 0 7 2
106 $god_ set-dist 0 8 3
107 $god_ set-dist 0 9 4
108 $god_ set-dist 0 10 4
109 $god_ set-dist 0 11 7
110 $god_ set-dist 0 12 4
111 $god_ set-dist 0 13 3
112 $god_ set-dist 0 14 1
113 $god_ set-dist 0 15 3
114 $god_ set-dist 0 16 3
115 $god_ set-dist 0 17 3
116 $god_ set-dist 0 18 4
117 $god_ set-dist 0 19 2
118 $god_ set-dist 0 20 3
119 $god_ set-dist 0 21 1
120 $god_ set-dist 0 22 3
121 $god_ set-dist 0 23 2
122 $god_ set-dist 0 24 2
123 $god_ set-dist 0 25 1
124 $god_ set-dist 0 26 4
125 $god_ set-dist 0 27 3
126 $god_ set-dist 0 28 16777215
127 $god_ set-dist 0 29 3
128 $god_ set-dist 0 30 2
129 $god_ set-dist 0 31 2
130 $god_ set-dist 0 32 3
```

```
$god_ set-dist 0 33 2
$god_ set-dist 0 34 5
$god_ set-dist 0 35 3
$god_ set-dist 0 36 5
$god_ set-dist 0 37 4
$god_ set-dist 0 38 4
$god_ set-dist 0 39 4
$god_ set-dist 0 40 1
$god_ set-dist 0 41 1
$god_ set-dist 0 42 3
$god_ set-dist 0 43 3
$god_ set-dist 0 44 1
$god_ set-dist 0 45 4
$god_ set-dist 0 46 6
$god_ set-dist 0 47 4
$god_ set-dist 0 48 4
$god_ set-dist 0 49 3
$god_ set-dist 0 50 2
$god_ set-dist 0 51 4
$god_ set-dist 0 52 2
$god_ set-dist 0 53 2
$god_ set-dist 0 54 7
$god_ set-dist 0 55 3
$god_ set-dist 0 56 4
$god_ set-dist 0 57 4
$god_ set-dist 0 58 4
$god_ set-dist 0 59 3
...
$ns_ at 10.000000000000 "$node_(0) setdest
27.629667531459 192.846042223269
6.310634010614"
$ns_ at 10.000000000000 "$node_(1) setdest
39.569285798448 723.766542608720
7.018073932212"
$ns_ at 10.000000000000 "$node_(2) setdest
233.468527407724 209.046721454178
7.068608305480"
$ns_ at 10.000000000000 "$node_(3) setdest
149.149441724430 895.991444929089
8.047025267450"
$ns_ at 10.000000000000 "$node_(4) setdest
879.140318455451 30.590344677398
9.485996857539"
$ns_ at 10.000000000000 "$node_(5) setdest
```

```
      722.554230804388 363.216263765698
      6.566607697367"
165   $ns_ at 10.000000000000 "$node_(6) setdest
      283.444564872531 360.476208380203
      3.914387436628"
166   $ns_ at 10.000000000000 "$node_(7) setdest
      685.632237654996 591.860382050965
      6.339151141826"
167   $ns_ at 10.000000000000 "$node_(8) setdest
      20.805033419152 685.344790539638
      4.600700008903"
168   $ns_ at 10.000000000000 "$node_(9) setdest
      271.030959597425 481.484328966760
      9.854185129439"
169   $ns_ at 10.000000000000 "$node_(10) setdest
      402.727477012043 547.652009970420
      4.314940352344"
170   $ns_ at 10.000000000000 "$node_(11) setdest
      662.219545627401 225.425769688692
      3.152239521888"
171   $ns_ at 10.000000000000 "$node_(12) setdest
      569.016818971256 898.353737978632
      4.542628304303"
172   $ns_ at 10.000000000000 "$node_(13) setdest
      485.857656937650 281.292626380191
      4.502535214770"
173   $ns_ at 10.000000000000 "$node_(14) setdest
      466.162846670922 758.624230277222
      5.189541373268"
174   $ns_ at 10.000000000000 "$node_(15) setdest
      511.607797843037 114.645114994185
      3.314395153288"
175   $ns_ at 10.000000000000 "$node_(16) setdest
      297.903253409052 115.235549274132
      8.511149752959"
176   $ns_ at 10.000000000000 "$node_(17) setdest
      862.385887297863 469.011961937530
      0.024160210748"
177   $ns_ at 10.000000000000 "$node_(18) setdest
      397.986192275101 776.804171147586
      8.316421445882"
178   $ns_ at 10.000000000000 "$node_(19) setdest
      875.893800795523 366.172366697307
      0.615854116179"
```

```
179 $ns_ at 10.000000000000 "$node_(20) setdest
    725.186024265991 457.612867067446
    8.858095624597"
180 ...
181 $ns_ at 10.153811401797 "$god_ set-dist 9 27
    2"
182 $ns_ at 10.153811401797 "$god_ set-dist 9 45
    3"
183 $ns_ at 10.153811401797 "$god_ set-dist 18 27
    1"
184 $ns_ at 10.153811401797 "$god_ set-dist 18 45
    2"
185 $ns_ at 10.153811401797 "$god_ set-dist 27 36
    3"
186 $ns_ at 10.153811401797 "$god_ set-dist 27 39
    2"
187 $ns_ at 10.153811401797 "$god_ set-dist 27 51
    2"
188 $ns_ at 10.153811401797 "$god_ set-dist 27 56
    2"
189 $ns_ at 10.153811401797 "$god_ set-dist 36 45
    4"
190 $ns_ at 10.153811401797 "$god_ set-dist 39 45
    3"
191 $ns_ at 10.153811401797 "$god_ set-dist 45 51
    3"
192 $ns_ at 10.153811401797 "$god_ set-dist 45 56
    3"
193 $ns_ at 11.120839557790 "$god_ set-dist 35 37
    4"
194 $ns_ at 11.120839557790 "$god_ set-dist 35 42
    2"
195 $ns_ at 11.120839557790 "$god_ set-dist 35 59
    3"
196 $ns_ at 11.158065266502 "$god_ set-dist 3 52
    5"
197 $ns_ at 11.158065266502 "$god_ set-dist 5 52
    3"
198 $ns_ at 11.158065266502 "$god_ set-dist 8 52
    1"
199 $ns_ at 11.158065266502 "$god_ set-dist 9 52
    2"
200 $ns_ at 11.158065266502 "$god_ set-dist 10 52
    2"
```

`

```
$ns_ at 11.158065266502 "$god_ set-dist 11 52
5"
$ns_ at 11.158065266502 "$god_ set-dist 16 52
2"
$ns_ at 11.158065266502 "$god_ set-dist 22 52
2"
$ns_ at 11.158065266502 "$god_ set-dist 26 52
4"
$ns_ at 11.158065266502 "$god_ set-dist 34 52
3"
$ns_ at 11.158065266502 "$god_ set-dist 36 52
3"
$ns_ at 11.158065266502 "$god_ set-dist 38 52
3"
...
#
# Destination Unreachables: 59
#
# Route Changes: 11104
#
# Link Changes: 2277
#
# Node | Route Changes | Link Changes
#    0 |           488 |           65
#    1 |           374 |           74
#    2 |           285 |           63
#    3 |           547 |          105
#    4 |           600 |           99
#    5 |           381 |           65
#    6 |           267 |           92
#    7 |           268 |           86
#    8 |           509 |           49
#    9 |           337 |           85
#   10 |           320 |           87
#   11 |           451 |           35
#   12 |           509 |           80
#   13 |           297 |           73
#   14 |           308 |           83
#   15 |           333 |          107
#   16 |           285 |           75
#   17 |           174 |           78
#   18 |           369 |           97
#   19 |           362 |           36
#   20 |           372 |          125
```

```
238 |  #    21 |              377 |               44
239 |  #    22 |              303 |               62
240 |  #    23 |              377 |               61
241 |  #    24 |              413 |              101
242 |  #    25 |              264 |               87
243 |  #    26 |              476 |               67
244 |  #    27 |              275 |               78
245 |  #    28 |              415 |               61
246 |  #    29 |              438 |               69
247 |  #    30 |              354 |               64
248 |  #    31 |              416 |               80
249 |  #    32 |              335 |               72
250 |  #    33 |              267 |               69
251 |  #    34 |              300 |               78
252 |  #    35 |              301 |               61
253 |  #    36 |              363 |              100
254 |  #    37 |              284 |               96
255 |  #    38 |              397 |               98
256 |  #    39 |              343 |               97
257 |  #    40 |              359 |               70
258 |  #    41 |              464 |               91
259 |  #    42 |              472 |               74
260 |  #    43 |              209 |               60
261 |  #    44 |              315 |               67
262 |  #    45 |              437 |               33
263 |  #    46 |              400 |               83
264 |  #    47 |              408 |               53
265 |  #    48 |              445 |              118
266 |  #    49 |              271 |              100
267 |  #    50 |              389 |               50
268 |  #    51 |              345 |               91
269 |  #    52 |              435 |               32
270 |  #    53 |              240 |               87
271 |  #    54 |              505 |               28
272 |  #    55 |              442 |              114
273 |  #    56 |              409 |               46
274 |  #    57 |              452 |               95
275 |  #    58 |              355 |               69
276 |  #    59 |              322 |               89
277 |  #
```

Gambar 8.1 Contoh skenario *node-movement* dari setdest

`

```
1   set val(chan)        Channel/WirelessChannel
2   set val(prop)        Propagation/Nakagami
3   set val(netif)       Phy/WirelessPhy
4   set val(mac)         Mac/802_11
5   set val(ifq)         CMUPriQueue
6   set val(ll)          LL
7   set val(ant)         Antenna/OmniAntenna
8   set opt(x)           900;
9   set opt(y)           900;
10  set val(ifqlen)      50;
11  set val(nn)          90;
12  set val(seed)        0.0
13  set val(adhocRouting)   DSR
14  set val(stop)           200
15  simulation time
16  set val(cp)          "cbrfix"
17
18  set val(sc)          "scenario-m10-n90-5"
19
20  Phy/WirelessPhy set RXThresh_ 0.71341e-08 ;
21  set ns_       [new Simulator]
22  set topo      [new Topography]
23
24
25  set tracefd  [open DSR_$val(sc).tr w]
26  set namtrace  [open DSR_$val(sc).nam w]
27
28  $ns_ trace-all $tracefd
29  $ns_ namtrace-all-wireless $namtrace $opt(x)
30  $opt(y)
31
32  set topo      [new Topography]
33  $topo load_flatgrid $opt(x) $opt(y)
34
35  set god_ [create-god $val(nn)]
36
37  $ns_ node-config -adhocRouting
38  $val(adhocRouting) \
39                   -llType $val(ll) \
40                   -macType $val(mac) \
41                   -ifqType $val(ifq) \
42                   -ifqLen $val(ifqlen) \
43                   -antType $val(ant) \
44                   -propType $val(prop) \
```

```
                -phyType $val(netif) \
               -channelType $val(chan) \
                       -topoInstance $topo \
                              -agentTrace ON \
                              -routerTrace ON \
                               -macTrace OFF \
                              -movementTrace ON \


 [$val(nn)] and "attach" them
for {set i 0} {$i < $val(nn)} {incr i} {
             set node_($i) [$ns_ node]
             $node_($i) random-motion 0 ;#
disable random motion
}

puts "Loading connection pattern..."
source $val(cp)

puts "Loading scenario file..."
source $val(sc)

for {set i 0} {$i < $val(nn)} {incr i} {
        $ns_ initial_node_pos $node_($i) 20
}
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $val(stop).0 "$node_($i) reset";
}

#$ns_ at  $val(stop)       "stop"
$ns_ at  $val(stop).0002 "puts \"NS
EXITING...\" ; $ns_ halt"

puts $tracefd "M 0.0 nn $val(nn) x $opt(x) y
$opt(y) rp $val(adhocRouting)"
puts $tracefd "M 0.0 sc $val(sc) cp $val(cp)
seed $val(seed)"
puts $tracefd "M 0.0 prop $val(prop) ant
$val(ant)"

puts "Starting Simulation..."
$ns_ run

```

Gambar 8.2 *Script* DSR.tcl

```
1   BEGIN {
2   sent = 0;
3   recv = 0;
4   pdr = 0;
5   }
6   {
7
8   if ($1 == "s" && $3 == "_1_" && $4 == "AGT" &&
9   $7 == "cbr")
10  {
11          sent++;
12  }
13
14  if ($1 == "r" && $3 == "_2_" && $4 == "AGT" &&
15  $7 == "cbr")
16  {
17          recv++;
18  }
19  }
20  END {
21  pdr = (recv/sent) * 100;
22  print ("Transmitted packet(s) : ", sent);
23  print ("Received packet(s) : ", recv);
24  print ("Packet Delivery Ratio : ", pdr, "%");
25  }
```

Gambar 8.3 Implementasi *Packet Delivery Ratio* (PDR)

```
1   BEGIN {
2   rt_paket = 0;
3   }
4   {
5   if (($1 == "s" || $1 == "f") && $4 == "RTR" &&
6   $7 == "DSR")
7   {
8           rt_paket++;
9   }
11  }
12  END {
13  print ("Jumlah Routing Overhead : ",
14  rt_paket);
15  }
```

Gambar 8.4 Implementasi *Routing Overhead* (RO)

```
1   BEGIN {
2   for (i in pkt_id){
3   pkt_id[i] = 0;
4   }
5   for (i in pkt_sent){
6         pkt_sent[i] = 0;
7   }
8   for (i in pkt_recv){
9         pkt_recv[i] = 0;
10  }
11  delay = avg_delay = 0;
12  recv = 0;
13  recv_id = 0;
14  }
15  {
16
17  if ($1 == "s" && $3 == "_1_" && $4 == "AGT" &&
18  $7 == "cbr")
19  {
20        pkt_sent[$6] = $2;
21  }
22
23  if ($1 == "r" && $3 == "_2_" && $4 == "AGT" &&
24  $7 == "cbr" &&  recv_id != $6 )
25  {
26        recv++;
27        recv_id = $6;
28        pkt_recv[$6] = $2;
29  }
30  }
31  END {
32  for (i in pkt_recv){
33        delay += pkt_recv[i] - pkt_sent[i];
34  }
35  avg_delay = delay / recv;
36  print ("Total Packet(s) Receive : ",recv);
37  print ("Total Delay : ",delay);
38  print ("Average Packet Delivery Delay :
39  ",avg_delay);
40  }
```

Gambar 8.5 Implementasi *End-to-End Delay*

```
1   s 2.556838879 _1_ AGT  --- 0 cbr 512 [0 0 0 0]
    ------- [1:0 2:0 32 0] [0] 0 2
2   r 2.556838879 _1_ RTR  --- 0 cbr 512 [0 0 0 0]
    ------- [1:0 2:0 32 0] [0] 0 2
3   s 2.560723875 _1_ RTR  --- 1 DSR 32 [0 0 0 0]
    ------- [1:255 2:255 32 0] 1 [1 1] [0 1 0 0-
    >0] [0 0 0 0->0]
4   r 2.561664050 _57_ RTR  --- 1 DSR 32 [0
    ffffffff 1 800] ------- [1:255 2:255 32 0] 1
    [1 1] [0 1 0 0->0] [0 0 0 0->0]
5   r 2.561664058 _38_ RTR  --- 1 DSR 32 [0
    ffffffff 1 800] ------- [1:255 2:255 32 0] 1
    [1 1] [0 1 0 0->0] [0 0 0 0->0]
6   r 2.561664166 _47_ RTR  --- 1 DSR 32 [0
    ffffffff 1 800] ------- [1:255 2:255 32 0] 1
    [1 1] [0 1 0 0->0] [0 0 0 0->0]
7   r 2.561664200 _10_ RTR  --- 1 DSR 32 [0
    ffffffff 1 800] ------- [1:255 2:255 32 0] 1
    [1 1] [0 1 0 0->0] [0 0 0 0->0]
8   r 2.561664250 _18_ RTR  --- 1 DSR 32 [0
    ffffffff 1 800] ------- [1:255 2:255 32 0] 1
    [1 1] [0 1 0 0->0] [0 0 0 0->0]
9   r 2.561664253 _22_ RTR  --- 1 DSR 32 [0
    ffffffff 1 800] ------- [1:255 2:255 32 0] 1
    [1 1] [0 1 0 0->0] [0 0 0 0->0]
10  r 2.561664262 _56_ RTR  --- 1 DSR 32 [0
    ffffffff 1 800] ------- [1:255 2:255 32 0] 1
    [1 1] [0 1 0 0->0] [0 0 0 0->0]
11  r 2.561664284 _51_ RTR  --- 1 DSR 32 [0
    ffffffff 1 800] ------- [1:255 2:255 32 0] 1
    [1 1] [0 1 0 0->0] [0 0 0 0->0]
12  r 2.561664287 _39_ RTR  --- 1 DSR 32 [0
    ffffffff 1 800] ------- [1:255 2:255 32 0] 1
    [1 1] [0 1 0 0->0] [0 0 0 0->0]M 10.00000 0
    (85.15, 869.15, 0.00), (27.63, 192.85), 6.31
13  M 10.00000 1 (507.27, 230.82, 0.00), (39.57,
    723.77), 7.02
14  M 10.00000 2 (226.25, 346.11, 0.00), (233.47,
    209.05), 7.07
15  M 10.00000 3 (874.74, 106.31, 0.00), (149.15,
    895.99), 8.05
16  M 10.00000 4 (255.92, 832.62, 0.00), (879.14,
    30.59), 9.49
17  M 10.00000 5 (505.10, 16.04, 0.00), (722.55,
```

```
     363.22), 6.57
18   M 10.00000 6 (696.76, 580.13, 0.00), (283.44,
     360.48), 3.91
19   M 10.00000 7 (317.60, 658.86, 0.00), (685.63,
     591.86), 6.34
20   M 10.00000 8 (221.59, 349.15, 0.00), (20.81,
     685.34), 4.60
21   M 10.00000 9 (324.26, 251.18, 0.00), (271.03,
     481.48), 9.85
22   M 10.00000 10 (425.78, 284.42, 0.00), (402.73,
     547.65), 4.31
23   M 10.00000 11 (894.89, 3.48, 0.00), (662.22,
     225.43), 3.15
24   M 10.00000 12 (864.47, 357.40, 0.00), (569.02,
     898.35), 4.54
25   M 10.00000 13 (624.06, 758.36, 0.00), (485.86,
     281.29), 4.50
26   M 10.00000 14 (106.25, 804.76, 0.00), (466.16,
     758.62), 5.19
27   M 10.00000 15 (446.86, 531.89, 0.00), (511.61,
     114.65), 3.31
28   M 10.00000 16 (446.83, 432.96, 0.00), (297.90,
     115.24), 8.51
29   M 10.00000 17 (487.54, 489.43, 0.00), (862.39,
     469.01), 0.02
30   M 10.00000 18 (529.43, 341.30, 0.00), (397.99,
     776.80), 8.32
31   M 10.00000 19 (10.97, 508.37, 0.00), (875.89,
     366.17), 0.62
32   f 56.374644855 _43_ RTR  --- 100 DSR 48 [0
     ffffffff 4 800] ------- [4:255 5:255 32 0] 2
     [1 2] [0 2 0 0->0] [0 0 0 0->0]
33   f 56.375790565 _6_ RTR  --- 100 DSR 48 [0
     ffffffff 4 800] ------- [4:255 5:255 32 0] 2
     [1 2] [0 2 0 0->0] [0 0 0 0->0]
34   r 56.375833199 _9_ RTR  --- 100 DSR 48 [0
     ffffffff 2b 800] ------- [4:255 5:255 32 0] 2
     [1 2] [0 2 0 0->0] [0 0 0 0->0]
35   r 56.375833212 _35_ RTR  --- 100 DSR 48 [0
     ffffffff 2b 800] ------- [4:255 5:255 32 0] 2
     [1 2] [0 2 0 0->0] [0 0 0 0->0]
36   r 56.375833212 _15_ RTR  --- 100 DSR 48 [0
     ffffffff 2b 800] ------- [4:255 5:255 32 0] 2
     [1 2] [0 2 0 0->0] [0 0 0 0->0]
```

`

```
37  r 56.375833263 _24_ RTR  --- 100 DSR 48 [0
    ffffffff 2b 800] ------- [4:255 5:255 32 0] 2
    [1 2] [0 2 0 0->0] [0 0 0 0->0]
38  r 56.375833293 _53_ RTR  --- 100 DSR 48 [0
    ffffffff 2b 800] ------- [4:255 5:255 32 0] 2
    [1 2] [0 2 0 0->0] [0 0 0 0->0]
39  r 56.375833303 _38_ RTR  --- 100 DSR 48 [0
    ffffffff 2b 800] ------- [4:255 5:255 32 0] 2
    [1 2] [0 2 0 0->0] [0 0 0 0->0]
40  r 56.375833305 _10_ RTR  --- 100 DSR 48 [0
    ffffffff 2b 800] ------- [4:255 5:255 32 0] 2
    [1 2] [0 2 0 0->0] [0 0 0 0->0]
41  r 56.375833318 _1_ RTR  --- 100 DSR 48 [0
    ffffffff 2b 800] ------- [4:255 5:255 32 0] 2
    [1 2] [0 2 0 0->0] [0 0 0 0->0]
42  r 56.375833324 _34_ RTR  --- 100 DSR 48 [0
    ffffffff 2b 800] ------- [4:255 5:255 32 0] 2
    [1 2] [0 2 0 0->0] [0 0 0 0->0]
43  r 56.375833373 _2_ RTR  --- 100 DSR 48 [0
    ffffffff 2b 800] ------- [4:255 5:255 32 0] 2
    [1 2] [0 2 0 0->0] [0 0 0 0->0]
44  r 56.375833381 _36_ RTR  --- 100 DSR 48 [0
    ffffffff 2b 800] ------- [4:255 5:255 32 0] 2
    [1 2] [0 2 0 0->0] [0 0 0 0->0]
45  r 56.375833387 _39_ RTR  --- 100 DSR 48 [0
    ffffffff 2b 800] ------- [4:255 5:255 32 0] 2
    [1 2] [0 2 0 0->0] [0 0 0 0->0]
46  r 56.375833460 _17_ RTR  --- 100 DSR 48 [0
    ffffffff 2b 800] ------- [4:255 5:255 32 0] 2
    [1 2] [0 2 0 0->0] [0 0 0 0->0]
47  r 56.375833517 _22_ RTR  --- 100 DSR 48 [0
    ffffffff 2b 800] ------- [4:255 5:255 32 0] 2
    [1 2] [0 2 0 0->0] [0 0 0 0->0]
48  r 56.375833541 _4_ RTR  --- 100 DSR 48 [0
    ffffffff 2b 800] ------- [4:255 5:255 32 0] 2
    [1 2] [0 2 0 0->0] [0 0 0 0->0]
49  f 56.376034913 _53_ RTR  --- 100 DSR 68 [0
    ffffffff 2b 800] ------- [4:255 5:255 32 0] 3
    [1 2] [0 2 0 0->0] [0 0 0 0->0]0 ffffffff 1
50  800] ------- [1:255 -1:255 30 0] [0x2 0 3 [2
    0] [1 8]] (REQUEST)

51  ...
```

```
     SFESTs 199.325328230 _4_ 840 [4 -> 5] 35(1) to
52   59 [4 |59 5 ]
     s 199.325328230 _4_ RTR  --- 840 cbr 552 [0 0
     0 0] ------- [4:0 5:0 32 59] [141] 0 1
53   r 199.331065315 _59_ RTR  --- 840 cbr 552 [13a
     3b 4 800] ------- [4:0 5:0 32 59] [141] 1 1
54   f 199.331065315 _59_ RTR  --- 840 cbr 552 [13a
     3b 4 800] ------- [4:0 5:0 31 5] [141] 1 1
55   r 199.337307124 _5_ RTR  --- 840 cbr 552 [13a
     5 3b 800] ------- [4:0 5:0 31 5] [141] 2 1
56   r 199.337307124 _5_ AGT  --- 840 cbr 512 [13a
     5 3b 800] ------- [4:0 5:0 31 5] [141] 2 1
57   D 200.000000000 _2_ IFQ  END 596 DSR 92 [0 0 2
     800] ------- [2:255 1:255 254 5] 8 [0 18] [1
     18 8 1->2] [0 0 0 0->0]
```

Gambar 8.6 Contoh *trace file* dari *script* DSR.tcl

BIODATA PENULIS

Hasbi As Shiddiqi, biasa dipanggil Hasbi, dilahirkan di Surabaya pada tanggal 08 Mei 1992. Penulis adalah anak pertama dari tiga bersaudara. Penulis menempuh TK Al-Ikhlas (1996-1998), SD Negeri Kendangsari 3 Surabaya (1998-2004), SMP Negeri 35 Surabaya (2004-2007), SMA Negeri 20 Surabaya (2007-2010). Pada tahun 2010, penulis memulai pendidikan S1 Jurusan Teknik Informatika Fakultas Teknologi Informasi di Institut Teknologi Sepuluh Nopember Surabaya yang terdaftar dengan NRP 5110100058. Di jurusan Teknik Informatika, penulis mengambil bidang minat Arsitektur dan Jaringan Komputer (AJK). Penulis dapat dihubungi melalui alamat *e-mail* hasbi.058@gmail.com.